NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

A PROPOSED ARCHITECTURE FOR ON-LINE JDISS TRAINING

by

William P. Kopper

September, 1996

Thesis Advisor: Suresh Sridhar

Approved for public release; distribution is unlimited.

19961220 112

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A PROPOSED ARCHITECTURE FOR ON-LINE JDISS TRAINING	5. FUNDING NUMBERS
6. AUTHOR(S) William P. Kopper	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The Joint Deployable Intelligence Support System (JDISS) is the primary Department of Defense system for the exchange of intelligence information. Unfortunately, the system lacks an adequate computer learning application. The widespread implementation of universally connected client/server networks, such as JDISS, will soon be enhanced by advanced World Wide Web (the Web) services. The combination of the Web and object-oriented technology, known as distributed object technology, may provide a solution to this problem. Use of the Common Object Request Brokerage Architecture (CORBA) Object Request Broker (ORB) could allow users of the (JDISS) to create customized training modules on-line the distributed object environment. Until JDISS adopts an ORB standard, an intermediary on-line training system based on advanced HTML, Java applets, and JavaScript could provide some of the functionality expected of an ORB based system. Regardless of the technology used to develop on-line JDISS training, certain system requirements must be met by any system to ensure its success. These requirements are defined from both the user and system administrator perspectives.

14. SUBJECT TERMS: CORBA, Intelligence Dissemination, Java, JavaScript, JDISS, Training, World Wide Web	15. NUMBER OF PAGES 116
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

A PROPOSED ARCHITECTURE FOR ON-LINE JDISS TRAINING

William P. Kopper
Lieutenant, United States Navy
B.A., Northwestern University, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the
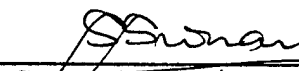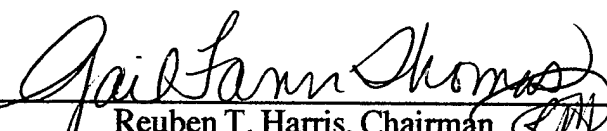
NAVAL POSTGRADUATE SCHOOL
September 1996

Author: _____
William P. Kopper

Approved by: _____
Suresh Sridhar, Thesis Advisor

_____
Tung Bui, Associate Advisor

_____
Reuben T. Harris, Chairman
Department of Systems Management

iii

ABSTRACT

The Joint Deployable Intelligence Support System (JDISS) is the primary Department of Defense system for the exchange of intelligence information. Unfortunately, the system lacks an adequate computer learning application. The widespread implementation of universally connected client/server networks, such as JDISS, will soon be enhanced by advanced World Wide Web (the Web) services. The combination of the Web and object-oriented technology, known as distributed object technology, may provide a solution to this problem. Use of the Common Object Request Brokerage Architecture (CORBA) Object Request Broker (ORB) could allow users of the (JDISS) to create customized training modules on-line the distributed object environment. Until JDISS adopts an ORB standard, an intermediary on-line training system based on advanced HTML, Java applets, and JavaScript could provide some of the functionality expected of an ORB based system. Regardless of the technology used to develop on-line JDISS training, certain system requirements must be met by any system to ensure its success. These requirements are defined from both the user and system administrator perspectives.

TABLE OF CONTENTS

x

I.   INTRODUCTION

The widespread implementation of universally connected client/server computer networks has changed the way information technology services are designed and delivered. The concept of a static embedded service available on a single machine is no longer common. The new model of computing focuses on accessing and sharing distributed services from any node on the network. The introduction of the Hyper Text Transfer Protocol (HTTP) and companion Hyper Text Markup Language (HTML) files has expanded information retrieval beyond the local area network (LAN) to any node on the World Wide Web (the Web) regardless of the type of hardware or operating system.

Soon the combination of object-oriented technology and networking, through middleware schemes such as the Common Object Request Brokerage Architecture (CORBA), will erase the difference between clients and servers. According to Orfali, Hackey, and Edwards (1996), objects, distributed on millions of machines, will behave like lego blocks with the ability to collaborate across the network in an infinite number of customized applications. This thesis will examine how training for the Joint Deployable Intelligence Support System (JDISS) can be delivered on-line via the Web in a way that takes advantage of today's client/server environment while

providing the necessary hooks for easy migration to tomorrow's distributed object architecture.

A. JDISS CONCEPT

JDISS provides hardware, software, and protocol standards for computer to computer exchange of sensitive intelligence information across the Department of Defense (DoD) and Allied communications networks. A typical JDISS node consists of a Unix workstation hosting a variety of Commercial/Government-off-the-shelf (COTS/GOTS) software components. Connection of the workstation to the appropriate military communications network allows for intelligence dissemination and interchange. The node owner determines which classified network (SIPRNET, DSNET-III, NATO, etc.) to access based on mission and security requirements. The JDISS Project Management Office (PMO) determines which services the system requires in response to sponsor and user feedback. It also consults with other PMO's to ensure interoperability between JDISS nodes and other systems connected to a given network. Due to its proven track record, JDISS has become the premier intelligence dissemination system with nearly 3500 nodes worldwide. (JDISS PMO 1995) Appendix A describes the authorities that govern JDISS configuration and evolution.

B. JDISS TRAINING

Unfortunately, JDISS training trails behind deployment. At times a JDISS system arrives at a unit only several days before or during a force deployment, without any required

training. Although most intelligence professionals are familiar with JDISS, they may not immediately be proficient with the system at all times. System upgrades further compound the training problem by providing improved services for which training may be necessary, but not available. Thus, JDISS users may have little or no familiarity with the system which will become their primary source of intelligence information. At this time the only training available outside the classroom is JDISS Embedded Support (JES), a multimedia service provided as part of all JDISS software suites. (JDISS 1996)

Replacing JES with a Web mechanism to deliver computer learning materials via the network will improve the following:

- Training availability
- Service tailoring
- System updates and re-fresher training

1.   Training Availability

Embedded training within the system in the form of JES 2.0 is static, out of date, and presented in an unfriendly, hard to update, format. Also, JDISS training is not readily available to users, unless they are near a JDISS equipped command. Web training could extend training availability through delivery over the unclassified Web to any student with a Web browser equipped PC. The same material could be etched into unclassified CD ROM disks for use on standalone systems.

2. Service Tailoring

Classroom courses are divided into two functional areas, administrator and operator. JDISS training courses can be tailored for a specific group, but most often they are presented from a standard syllabus without taking user needs or experience into account. Web training would allow network delivered computer learning. In turn, this training could be structured to permit the student to determine which material to study based on his knowledge of the topic or his score on a quiz. Customized modules available on-line 24 hours a day let the student to learn at his own pace.

3. System Updates/Re-fresher Training

Users often lack the skills to use new or upgraded applications. A networked computer learning architecture would be easy to upgrade. Web pages can be updated at one location and accessed by multiple users. The very nature of the Web permits for quick editing of Web pages with readily available HTML editors such as Netscape Gold. (Netscape Gold 1996)

C. THESIS OBJECTIVES

This thesis provides a plan to expand JDISS training services to include Web based learning application available to all users. It will examine the concepts of Distance Learning, Computer Based Training, and Computer Conferencing. It will suggest how current and future JDISS configurations can support continuing education. Proposals for the Web

4

learning mechanism follow a migratory system improvement approach, ensuring the proposed JDISS training delivery plan meshes with current client/server and future distributed object architectures. Implementation will be network centric, relying on the Internet or the classified IntelNet for delivery and swift updates. Users will access information via commonly available Web browsers. Finally, a JDISS training delivery prototype will be described. The prototype delivers platform independent training over the network in the form of HTML documents enhanced with JavaScript and the object-oriented Java programming language.

D. THESIS ORGANIZATION

Chapter II will discuss the state of computer learning, examining various characteristics of learning and technologies that complement their implementation. Chapter III describes what the ultimate JDISS computer learning system would look like and outlines user and system requirements. Chapter IV explains how the coming convergence of network and object-oriented technologies influences the recommended architecture for JDISS Web training. Chapter V documents the implementation of an on-line JDISS training proof of concept. Lessons learned are provided in Chapter VI, while recommendations for further research and conclusions will be presented in Chapter VII.

II. COMPUTER LEARNING

Since its inception, Information Technology managers have
tried to implement the use of the Personal Computer (PC) into
their training plans, often with little to show for their
effort and expenditure. Allessi and Trollip (1985) assert
this failure has been the result of three shortcomings:

- Location of and access to the training laboratory
- Courseware selection
- Computer literacy of instructor and learner

However, according to Reinhardt (1995) the common
availability of three key technologies; Networked LANs,
Internet Protocols, and multimedia are making the idea of
widespread computer learning practical. Not only will such
education be available at any time during the day, it will be
available from any place the student is able to plug into the
network.

Additionally, the introduction of new schemes for
computer learning which emphasize self-paced, user driven
instruction will allow the user to be educated when he needs
the instruction and has a stake, task completion, in ensuring
that he learns properly. "Just-in-time" or "just-in-case"
training will allow organizations to link training to
productivity instead of providing education in advance of a
task. Several factors drive this approach to learning. The
first is the lack of resources applied toward training.

Second is the dispersion and turnover of employees. The third is the rapid evolution of technology. Combined these factors dictate that training must no longer be composed of static lessons presented in a traditional class room setting. (Reinhardt 1995)

Finally, computer learning allows the concept of one on one instruction, through apprenticeship, to be re-introduced back into training and education. Unlike the classroom where one expert teaches many students, computer access allows one student direct contact with hundreds of digital and live instructors. Instructors can become more like coaches, checking in on students progress and answering queries when the student is stuck. The student is also allowed greater flexibility to learn at his own pace, when he wishes to learn. (Reinhardt 1995) An examination of current thought concerning computer instruction, learning methodologies and supporting technologies provides an idea of what an on-line JDISS training system should be able to provide.

A. INSTRUCTION

Computer learning must support a variety of instructional methods if it is to be truly successful. According to Lauzon

and Moore (1992) and Kowitz and Smith (1987) learning occurs
in three phases:

- Learning the basics
- Using basic skills to acquire technical abilities
- Seeking to master the leading edge of knowledge and
  advance it

Within each form of instruction two dimensions can be
measured, density of the material to be learned and the amount
of human interaction necessary to impart that content. Kowitz
and Smith (1987) assert that throughout the course of learning
a subject, the student will choose ever increasing content
density while decreasing the amount of human interaction.

While learning the basics the student has very little
knowledge of the topic. Learning is controlled by the
instructor, a master of the subject. The instructor provides
the basic tools, schemas, and methodologies necessary to
master the skill or topic and evaluates the student's
progress. In the second phase of instruction, the learner,
who is skilled in the basics, begins to determine the
direction of study. Acting with the instructor, he acquires
more specialized skills and knowledge of the topic. The
teacher still evaluates the student, but evaluation process is
more a collaboration between the teacher and student. The
final phase of instruction is determined solely by the
student. By this time the learner is skilled in the
discipline and seeks to expand the field of knowledge through

9

his own analysis and interpretation of information. There is no teacher at this level of instruction. Instead, the learner relies on collaboration with other experts to expand his understanding of the topic and add to it. (Kowitz and Smith 1987)

Allesi and Trollip (1985) further maintain that four instructional methodologies exist which can be used to deliver training via networked computer systems. They are:

- Tutorials

- Drills and simulations

- Games

- Tests

These four methods can be used as the framework for training delivery for the first and second forms of instruction, learning basics and using basics to expand knowledge, which require an instructor to lead and monitor the student. These instruction methodologies will be considered as ways to present learning modules to students desiring on-line JDISS training. (Kowitz and Smith 1987)

With access to multiple levels of instruction delivered in a variety of formats to the student whenever and wherever he may be, a new kind of education, learner centered, becomes possible. According to Jackson, Straford, Krajcik, and Soloway (1996), such an approach allows each user to address his individual needs via computer learning. Under this scheme

the learner can design his own training model by combining various objects.

The method begins by building on the student's experience and prior knowledge of the topic. Through a series of interactive queries, the students can define the objects they wish to examine and provide the system with inputs which will be used to determine the level of knowledge the student has. Next new concepts are provided to the student based on the results of the queries. At last a series of "coupling actions" reinforces the lesson learned during the session by measuring the learners feedback and answers to questions. (Jackson, Straford, Krajcik, and Soloway 1996)

Guzdial and Kolodner (1996) explain that these techniques are similar to building a scaffold. The student starts by defining his base and builds a scaffold up from it as he learns more. Simply put, learning results from a process of complex problem solving combined with reflection. By measuring response to the problem, the proper case studies are delivered to the student. Thus, the student is ensured that he receives information on both the topic he is interested in and information that will increase his knowledge of that topic.

B.   TYPES OF COMPUTER INSTRUCTION

Three concepts interact to provide training and education to remote users via computers: Distance Learning (DL), Computer Based Training (CBT), and Computer Conferencing (CC).

11

While each started as a separate discipline using differing technology, the introduction of networks, multimedia, and Internet access have allowed these three fields to evolve into overlapping, interconnected ways of delivering instruction. These technologies have made the boundaries between each type of instruction nearly transparent. However, it is important to recognize the differences between them at an abstract level in order to design the proper training modules for Web learners. The concepts used to design Web training systems of the future will borrow from the tenets of all three ideas while relying on multiple formats for delivery. The educational schemes and delivery methods will depend on a hybrid approach, drawing on whichever available module of training or delivery medium best suits the need of the user.

1. Distance Learning

Initially, Distance Learning focused on delivering traditional class room lectures to students at distributed sights. As such, it originally utilized synchronous analog audio and video beamed to specialized video-teleconferencing (VTC) rooms, terminals, or telephones. Traditional distance learning often suffered for an assortment of reasons. First the instructor could not see or interact with his distant students. Second, although learning could be extended to remote locations, this often required special equipment and supporting conference rooms that limited the number of people who could participate. Special scheduling procedures limited

12

the availability of training and time slots available for learning. Finally, band width requirements often restricted the quality of instruction (particularly video) received.

Due to the advent of networked computers and ever increasing band width, Shackleton and Clark (1995) claim that numerous benefits can now be derived from using distance learning. Pupils can choose the environment and pace at which they study, while the instructor can support many levels and complexities of instruction. Through the use of audio conferencing and VTC, the organization can extend education beyond the classroom to multiple remote locales or even the desktop. The rapidly emerging availability of audio and VTC over LANs and the Internet now let information technology managers to extend remote computer learning to the desk top.

2. Computer Based Training

Computer Based Training is "an interactive learning experience between a learner and a computer in which the computer provides the majority stimulus, the learner must respond and the computer analyzes the response and provides feedback to the learner." (Gery 1991, p 6) The computer replaces the instructor and engages the user through tutorials, drills, and tests. With smart programming, student input may be measured to ensure the proper materials are delivered to build upon instruction based on response, leading to guided learning by the user.

Traditionally this type of instruction has been embedded as an educational or tutorial application, relying primarily on the presentation of text and graphics by the application and typed responses by the student. The introduction of multimedia via CD ROM, LAN, or the Internet has beefed up the ability of CBT to deliver a much more inter-active, interesting, and compelling form of instruction. (Weiland 1995)

3. Computer Conferencing

Computer Conferencing is the use of networked computers to exchange information. Common Computer Conferencing methods include e-mail, file transfer protocol, news, and mail groups. Computer Conferencing provides two advantages for student and teacher alike. First, the ability of networked users to exchange information over the Web has lead to the creation of virtual communities which allow for asynchronous transaction between teacher and student. The participants in a learning venture do not have to be interacting with the network simultaneously. Second, asynchronous access to learning resources allows for the creation of a virtual classroom which can be entered 24 hours a day from any node on the network. Students can download lecture notes from a prior day and post questions back to the instructor who can respond via e-mail, chatter, or VTC. Finally, the introduction of multi-media

14

over the network allows for richly enhanced files such as images or video to be delivered via computer conferencing. (Lauzon and Moore 1992)

C. SUPPORTING TECHNOLOGIES

Four computing technologies: networking, Internet access, Standardized General Markup Language (SGML), and multimedia have allowed DL, CBT, and CC to become the supporting concepts for the larger scheme of on-line, on demand computer learning.

1. Networking

Networking allows for the sharing of information between computers connected to the network. Without networking, computer learning would be limited to individual applications loaded on to standalone computers. As a networked system, JDISS clients will be able to access the services provided by training servers located locally or around the world.

2. TCP/IP

Internet access and its supporting TCP/IP protocols allow for the delivery and retrieval of information across disparate networks. Without Internet Protocols, delivery of educational material between local area networks would be difficult, if not impossible unless all networks adhered to the same software and hardware standards. Most import of the TCP/IP protocols for delivering training materials is HTTP. HTTP allows all clients and servers on the network to request and send Web documents written in the HTML.

3. SGML

SGML is the super set of HTML. It allows an author to create a file that contains data and instructions for the presentation of that data. This is achieved by marking up the text. Using SGML, an author will designate the structure of the document, identifying paragraphs, titles, heading and key words with markup tags collectively called a Document Type Definition (DTD). When finished he can designate the look of the document without changing the text. Additionally searches can be made on individual parts of the document. In an object-oriented environment this will allow the creation of customized documents (training packages) based on the student's query.

For example, a smart system could sense a low bandwidth connection and deliver the document in a format compatible with that medium. Once delivered the student may wish to apply a small font to the text and print it out as a handy reference card. The information can be implemented in a variety of formats without damaging its integrity. The down side of SGML is that tagging a document is labor intensive, like climbing a mountain. Just as the mountaineer must check every step and expend lots of energy when climbing the slope, every paragraph, heading, and item of interest in the DTD must be marked with the proper tag for SGML to work. This could become an arduous process if the document requires detailed tagging. However, once tagged, producing the a customized

16

document is like walking down the slope. The document, like the path of descent, is well known and tagged. The user simply designates the style the document should have.

4. Multimedia

Multimedia will allow the concepts of DL, CBT, and CC to be delivered to the desktop in an infinite number of combinations. With multimedia, the user can access information in text, graphic, audio, or video format or some combination thereof. Multimedia products can be delivered to a desktop by LAN or CD ROM if the computer has the proper sound and video card hardware configuration. Still another way is to provide these services to all users with network access via the Web and TCP/IP protocols. (McMahan 1995)

According to Gallaghan (1994) multimedia allows the computer to become an Electronic Performance Support System (EPSS). An EPSS system allows the intersecting technologies of multimedia to deliver just-in-time training to a student. Multimedia provides a computer with the ability to "model, represent, structure, and implement (EPSS) support electronically." (Gery 1991)

In such an environment, a student will not only be able to access training through a computer, he will also be able to designate the delivery media for the instruction. For instance, the student could initially study a training module by accessing a text and graphically enhanced HTML file. A live or recorded VTC could be used to reinforce the module

17

objectives. Finally, the student could use a VTC, e-mail, or collaborative white board to contact a human instructor to ask questions or pass test results.

III. SYSTEM REQUIREMENTS

A. JDISS COMPUTER LEARNING SYSTEM

The ultimate on-line training system would provide an easily navigable environment for learning JDISS capabilities. During the training session the system would generate a customized course of instruction based on test results, known user preferences, or other input. Additionally, the material would be tailored for presentation in a format based on those preferences .

The user interface must be simple; its proper use inherent to anyone familiar with the operation of a PC. The emphasis should be on learning JDISS, not learning how to use the JDISS training system. Sitting down for on-line JDISS system training should be like driving your car to a new destination with a good map. Although you have never been to the destination, the map will ensure you arrive safely and on time because it is well detailed with clear directions. Since you are driving a car you do not even have to worry about learning how it works. You already know how to drive. Thus, the best user interface would be the one every computer user knows; point and click access with a mouse.

Finally, the methods or applications invoked by the system should be transparent. The student should receive what he needs to best master JDISS and not have to worry how the

system provides training. An emphasis on three factors will ensure that such a JDISS training system is eventually achieved. They are:

- An understanding of system requirements
- A well planned marriage of the requirements to technology
- Designing the lessons in a way such that it is easy to use and draws in the student

B. STRUCTURE AND SYSTEM REQUIREMENTS

Since JDISS supports diverse military, interdiction, and humanitarian efforts, the user base is constantly changing. Additionally, JDISS is a relatively new system, the role and use of it within the organization is evolving on a parallel track with operational deployment. Before developing the system requirements, a model of the structure of military intelligence organizations has been generated. This model, first suggested by Conway (1968) serves as the framework upon which system requirements are built.

According to Conway (1968) and Goldberg and Rubin (1995) a system will tend to have a structure that parallels the organization which develops and uses it. Each part of a system can be broken down into smaller interconnected subsystems. These subsystems communicate with one another through interfaces. Again, these systems, subsystems, and

interfaces reflect the design and lines of communication within the organization. (Conway 1968, Mowbray and Zahavi 1995)

For example, most intelligence professionals operate in a dynamic environment, often in support of a Joint Task Force (JTF). The JTF Intelligence team pieces are plugged and unplugged according to commander's intent and the availability of forces to meet his requirements. As such the functions of JTF Intelligence may be dispersed over a large area and composed of members of all the services, national agencies, and allied forces, yet they must always be able to exchange information with one another.

Applying Conway's theory to the JTF suggests that each component should have a corresponding systems and sub-systems to help it perform its particular mission. Such a scheme should reflect that JTFs, by their nature, are confederations working together only until their mission is complete. The supporting systems of each component should have interfaces that facilitate easy linking and de-linking to other systems within the confederation. The goal of providing customized and up-to-date training to a rapidly changing group such as a JTF intelligence staff can be achieved using the JDISS system to deliver object-oriented components via the Web or its DoD classified version, the IntelNet.

1. GENERIC SYSTEM REQUIREMENTS

A System Enhancement Process (SEP) was used to develop the requirements for the JDISS training architecture. The proposed architecture will add modifications to the current JDISS system which will provide new functionality. Additionally, the future CORBA based architecture of the JDISS is considered to ensure this plan will be compatible with expected improvements. Finally, the proposal is designed with an eye on providing these enhancements to other components of JDISS. (Goldberg and Rubin 1995)

Any Web based JDISS service must be supported by an architecture which will allow the application to achieve the following:

- Platform independence
- Delivery to a remote, distributed user base
- Multimedia
- Rapid updating capabilities
- Component re-use
- Compatibility with future system architectures

Such flexibility will allow JDISS training to be provided to a diverse, rapidly changing user base while remaining compliant with the constantly improving overall architecture of the system.

a. Platform Independence

Today, most JDISS nodes employ Solaris or another Unix operating system. However, plans to port the system to

22

Windows NT have been drawn up. Since training is an unclassified component of the JDISS system, platform independence will allow system users to access training beyond the scope of the classified networks via unclassified mirror servers. This would also allow the JDISS system to be designated for operational support, while the intelligence specialist could train in non-secured spaces or even at home on his PC.

b. Delivery to a Distributed User Base

JDISS already has a widely distributed on-line base due the overwhelming popularity of the system and the ease of plugging into intelligence communication networks. With nearly 3500 users world wide, training must be available to the most remote of foreign locations while taking available bandwidth into account. On-line delivery will allow for self-paced and tailored learning anywhere on the network, 24 hours a day. If a terminal can not be made available for training, the student will be able to train on a standalone machine with a CD ROM drive. Web browser technology will meet all the criteria of this requirement and provided services for a variety of operating systems.

c. Multimedia

JDISS workstations are equipped for multimedia. Multimedia should be added to training materials to draw in the student, enhancing his learning experience and keeping him engaged. In many cases band width exists to support delivery

of multimedia files or formatted pages. When the network is not available or robust enough to support multimedia training, CD ROM technology can fill this shortcoming.

d. Rapid Updating Capabilities

Since the capabilities of JDISS change every few months, the ability to rapidly update training modules by modifying current components to deliver new instruction without re-programming the delivery application is imperative. To solve shortcomings in the current JDISS training application, JES, the delivery architecture should be re-designed to make customized up-to-date training available over the JDISS network to all users of the system. Updates should be performed at designated education sites. By doing so, the architecture may cease to simply be a delivery service. Searches of a central location will begin to help the student make a decision as to which training aid he needs to study.

e. Component Re-use

While instructional data for JDISS training may change, the applications for enhancing their delivery will not change as fast. By adopting object-oriented technology, only those objects within the training system which have changed will have to be updated. The complete training system will not have to be re-compiled. Additionally the same data could be presented in a variety of formats by applying the methods of different objects to that data.

f. Compatibility with Future Architectures

Finally, the training system must easily migrate with other JDISS services to a distributed object architecture. In the future, smart middleware will make the delivery route of customized training modules transparent and platform independent to the remote user. In fact the concept of remote user might even disappear. A distributed, object based system would support all users connected to the network regardless of location or hardware/software configuration. Additionally, an object based training architecture provides a parallel structure of objects which could be plugged and unplugged in order to support the training needs of such a diverse group of users. Ensuring compliance with CORBA would allow for easy connection of diverse resources without knowledge of where they are located or what they look like. (Orfali, Hackey, and Edwards, 1996)

2. TRAINING SPECIFIC REQUIREMENTS

Any Web based JDISS training service must be supported by
an architecture which will allow the application to perform
the following functions:

- Module selection

- Interactive training access

- Grading mechanism

- Training record maintenance

- Version control

These functions are training oriented services, necessary
for the proper administration of a modular, object-oriented
training application.

a. Module Selection

Before beginning to train, the user must have a way
to designate goals and preferences. Comparing these
selections to the available modules will allow the user to
select and receive those instructional elements which best
suit his needs. This type of service, provided by query based
matching, could be achieved through use of data base front
ends and Decision Support System (DSS) techniques.

b. Interactive Training Access

Despite the best efforts of educators, system
designers, and programmers, not all training will result from
completion of various lessons. Some interaction with human
experts may be necessary for clarification of the lesson.
Thus, the student must have a way to query a JDISS expert

while studying. VTC, e-mail, or telephony could provide access to the expert. Whichever method is used, access must be achieved within the training environment through the click of a mouse on an icon.

c. Grading Mechanism

Student progress must be measured in order to assess which training module, if any, should be studied. Grading also allows accrediting authorities to determine if qualifications can be granted to the student. Such grading must have a secure mechanism that ensures answer and score integrity. Additionally self-grading can allow the student to determine if he has mastered a subject and is ready to apply for formal recognition of his progress.

d. Training Record Maintenance

Once a student has completed course work, submitted it for grading, and received the qualification to use part of a system, this information must be recorded. Results can be used to update professional records and ensure that the student is not subjected to repetitive and unnecessary training. The date of training completion can be recorded and marked in order to notify the student when refresher training may be needed.

e. Version Control and Module Management

Since the system is to provide multiple ways for the student to learn the same information, a scheme for version control and module management is necessary to ensure

the student receives the best lessons. Additionally, outdated material should be archived for historical reference.

In conclusion, a system employing the above principles will provide the backbone for customized, self-paced training in a collaborative environment. Creating a JDISS training delivery architecture through utilization of the latest object-oriented technology would leverage already available training materials by extending their access. Such a service can be designed to meet today's standards while acknowledging the future migration from the client/server environment to the distributed object model of tomorrow. Such a scheme would increase intelligence readiness through training and serve as a model for extending distributed object technology throughout DoDIIS and the DoD.

IV. JDISS TRAINING ARCHITECTURE AND SUPPORTING TECHNOLOGY

The previously recommended system architecture and supporting technology can be used as a means to implement the requirements of an on-line JDISS training system. The following technologies and methods are suggested following a study of emerging trends within the field of information technology and the known path of future JDISS evolution. A combination of Web browser, object-oriented, and distributed object technology will be necessary to build the perfect on-line JDISS training system on the foundation of the current JDISS infrastructure.

A. INTERNET DELIVERY

Flynn (1996) claims computing will undergo an Internet revolution "where distributed secure, platform independent software objects (will) provide users with gratification on demand and developers with unprecedented flexibility and power."

As a networked system, JDISS can participate in this network centric revolution. Specifically training materials can be delivered via the classified IntelLink or unclassified Internet. On-line delivery will allow for quick updates and 24 hour access to training materials. Training could be extended to anyone with a network connection and a Web browser, be they in the office or home, regardless of computer

29

type. The introduction of network enhancing middleware will change the concept of application from that of a static embedded program, replacing it with a dynamically created, customized application designed for a specific purpose.

B.   OBJECT-ORIENTED TECHNOLOGY

Object-oriented programming methodology provides a model to design a system of loosely confederated components capable of combining and breaking apart in an infinite number of combinations, just like a set of lego blocks. Objects are bits of code which contain behavior and provide their services through methods. The object code describes a set of behaviors and ways to implement those behaviors. The methods operate on private data called instance data owned by the object. Communications between objects are achieved by passing messages between their public interfaces. A collection of similar objects make up a class. A class describes the behavior of similar objects. (Orfali, Harkey, and Edwards 1996)

Three characteristics allow objects to remain independent
and combine with other objects:

- Encapsulation - the ability to hide methods from
  other objects and thus avoid being corrupted by them
- Inheritance - the ability to implement all of the
  methods of a parent of super-class
- Polymorphism - the ability of a method to do
  different things, depending on which class of object
  implements it

These features allow objects developed in the same programming
environment to be re-used in a near infinite variety of
configurations.

Numerous experts (Goldberg and Rubin 1995, Budd 1991,
Booch 1991) suggest object-oriented technology is the best way
to design information systems and software. Object-oriented
technologies allow the designer to model the system at a level
of abstraction similar to that of the real world. Each
process, node, activity, data set, or other entity in the
physical realm can be designated as a specific object,
allowing designers, programmers, and end users to describe
object behavior based on its physical counterpart.

Goldberg and Rubin (1995) go on to assert that the state
of object technology is such that it allows for rapid creation
of easily maintainable systems. The abstract properties of
objects allow one-to-one mapping of descriptions of physical
entities to object code. Objects can cooperate with one

another in a variety of ways, just as their real world counterparts. They can be combined into components to make an application and the same objects may be re-used later in a different configuration to create another application. (Budd 1991)

C. DISTRIBUTED OBJECT MIDDLEWARE AND CORBA

Middleware is a type of software which attempts to bridge the differences between computer languages and operating systems in order to facilitate cross environment cooperation. Distributed object middleware seeks to bridge these differences over nodes connected to a network or the Web. According to Korzeniowski (1995), distributed object middleware will allow programmers and users to combine objects over a distributed network and create components by mixing and matching objects regardless of their location and implementation. Smart middleware agents or brokers will be able to locate bits of code throughout the Web and bring them together for cooperation. Kador (1996) predicts access to the Web will become the preferred way of computing. (Kador 1996, Korzeniowski 1995)

Numerous experts (Mowbray and Zahavi 1995, Orfali, Hackey, and Edwards 1996, Tibbitts 1995, Chappell 1995) predict the Common Object Request Brokerage Architecture (CORBA) will become the standard for adding objects to the client/server environment. CORBA is the standard for a distributed object management architecture created by the

32

Object Management Group (OMG), a 600 member consortium of technology providers and vendors. (Tibbitts 1995) While other standards such as Microsoft ActiveX also seek to allow objects to communicate over networks, the strong links between SunSoft and the OMG, and the OMG's efforts to build bridges between CORBA and Active-X, suggest that JDISS will probably incorporate a CORBA backbone in the future. (OMG 1996b, Tibbits 1995, Korzeniowski 1995, Halfhill and Salamone 1996)

COBRA allows objects distributed over a network to communicate and collaborate regardless of implementation language or operating system. Orfali, Harkey and Edwards (1996, p 16) refer to this environment as the "intergalactic network" and predict that CORBA objects will be able to "run on different platforms, coexist with legacy applications through object wrappers, run on networks, and manage themselves and the resources they control." The Object Request Broker (ORB) provides the "common platform" for client objects to request services from server objects that reside on other nodes of the network. (Halfhill and Salamone 1996, p 103) The ORB furnishes object facilities and services which allow the transaction between client and server objects to appear transparent. Service objects collectively called object services extend the capability of the ORB, standardizing the way objects manage such properties as name,

version, life-cycle and security throughout a CORBA compliant network. (Halfhill and Salamone 1996, Orfali and Harkey 1995, Mowbray and Zahavi 1995)

Common facility objects provide uniform rules of engagement for objects to cooperate. These objects fall into two categories: vertical common facilities and horizontal common facilities. Horizontal facilities provide guidance for user interface, systems management, information management, and task management. The user interface facility governs the on screen services such as editing and window creation. Systems management objects provide interfaces for object installation, configuration, operation and repair. The task management services include work flow, scripting, agent, and transaction rules.

The information management objects define storage and data exchange. Unlike horizontal facilities which provide common rules that all ORBs must adhere to, vertical facilities provide the rules for specific industries or activities on a local ORB. Vertical facility objects might enforce rules for finance, marketing, or, in the case of JDISS, training rules. (Halfhill and Salamone 1996, Orfali and Harkey 1995, Mowbray and Zahavi 1995)

Finally, CORBA allows customized application objects to interact with the ORB. These objects are the application specific lego blocks from which customized services are built over the network. Any application object can interact with

the service and facility objects and other application objects as long as it has an Interface Definition Language (IDL) interface. IDL describes the interface on the object such that it is free of any association with a particular OS or programming language. Any ORB can understand the IDL interface of an object. The IDL interface understands the language of the object. Legacy applications may be wrapped in an IDL interface to allow for their use in a CORBA environment. The combination of IDL, CORBA facilities, and CORBA services allows objects to cooperate seamlessly be they on the same machine or half way around the world. Objects on a client will be able to ask objects on the server to perform a service for it, even if the client does not know where the server object resides. (Orfali, Harkey, and Edwards 1996, Soley 1995)

The CORBA 2.0 standard defines cooperation between ORB's. An object adapter on each ORB implements CORBA objects on the local ORB when its services are requested. The ORB knows whether it can respond to a request for a service by checking the Implementation Repository which documents the behavior of local objects. (Orfali, Harkey, and Edwards 1996) ORBs communicate over the Internet with the Internet Inter-ORB Protocol (IIOP), a TCP/IP compliant protocol. (Rymer 1995) CORBA 2.0 products should be widely implemented by 1997. Figure 1 illustrates how CORBA components are connected.
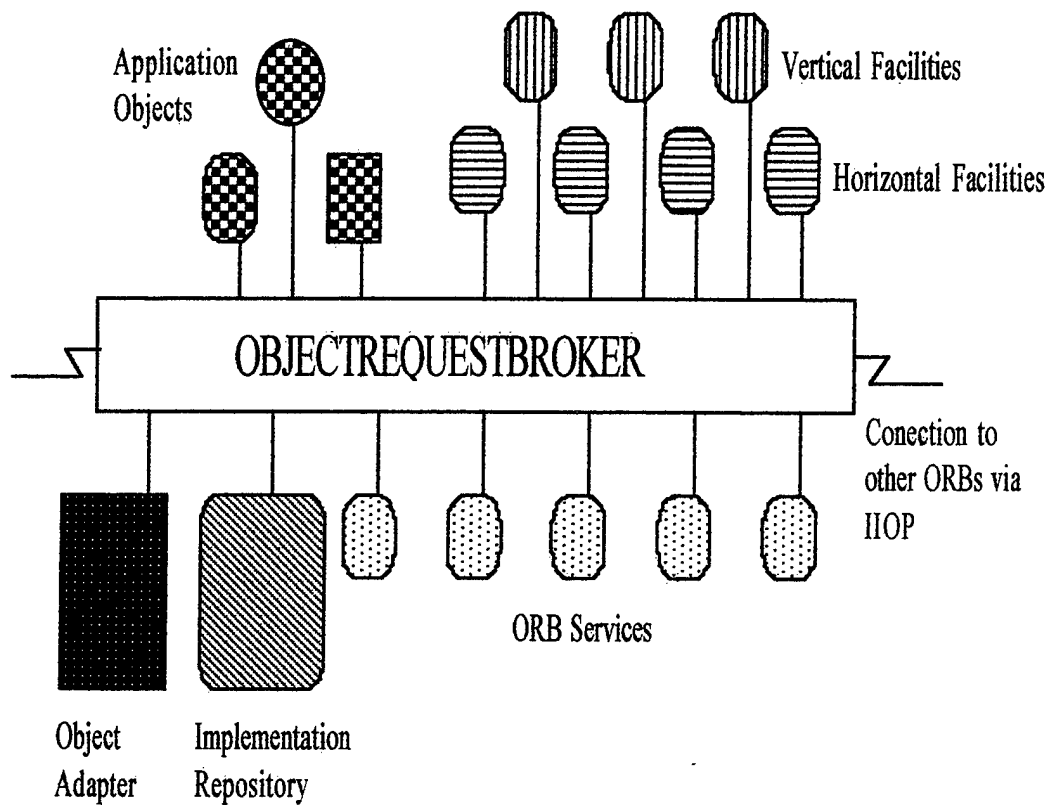
Figure 1. How CORBA Works.
After Orfali, Harkey, and Edwards 1996, p 54.

By extending the CORBA paradigm to Web training, it is possible to envision a future when various education objects and facilities collaborate to produce a customized training application for the user. The CORBA architecture also provides a common architecture for systems that may have never operated together. By using CORBA, new training modules can be plugged and unplugged whenever they are needed. They may be part of the network, invisible to the user, until application methods are called upon or they may come from new servers as they plug into a larger JTF type environment.

D. A PROPOSED MIGRATION SYSTEM

In order to implement the system requirements outlined in Chapter III with the technologies described above, a system based on the following technologies is recommended:

- Netscape Navigator and Servers
- Java and JavaScript programming languages

Use of these technologies will ensure that any training products created in the near future will be able to migrate to a pure CORBA or distributed object environment.

In the migration system, training materials will be delivered via a Web browser interface. The materials will be written in HTML and enhanced with Java applets and JavaScript scripts. While CORBA products are not yet readily available,

this architecture will support an easy migration to the distributed object architecture when the JDISS PMO implements it.

Utilizing the JDISS Multimedia Collaboration Manager (MCM), the student will be able to participate in VTC lectures and conferences with live instructors. Enhanced Web pages will compliment live instruction. In addition, the student will be able to take on-line quizzes embedded in HTML pages to measure his knowledge of the system. These quizzes will be automatically graded and recommendations for further study, including the Internet links to Web pages containing review material, will be presented to the student as his customized training application.

1. Netscape Navigator

According to Flynn (1996), three factors should guide the choice of a Web browser:

- Language and development tools
- Server platform
- Ability to extend capabilities of Web clients

Today, the Netscape Navigator, version 3.0, browser provides the most robust technology to satisfy the user. Netscape has led the way in browser technology and is often the first to introduce new additions. Quick advances and an aggressive pricing policy have allowed the Navigator browser to claim between 75 and 85% of the browser market. (Flynn 1996, p 29)

The Netscape browser supports HTML, Java and JavaScript. Often Netscape has implemented new extensions of HTML such as frames and tables. Also, the browser has an optional HTML editor. Finally, close cooperation with SunSoft has led to the addition of the Java virtual machine in version 2.0 and above. (SunSoft 1996)

JDISS does not currently employ any specific server product for delivery of its services. Each JDISS machine has all applications embedded. However, the introduction of Web training would represent the first JDISS service to be delivered from a server. As such, development of a server architecture could enhance delivery and availability of training services. One solution for enhanced dispatch of service might be implementation of the Netscape proxy server. The Netscape proxy server provides two important features which would be beneficial to providing Web training to a widely distributed client base with varying bandwidth capabilities. The first is replication. Replication allows the same HTML file, such as a new or updated training module, to be loaded on multiple servers automatically whenever it is updated. Once the new file is loaded on one server, it sends instructions to other proxy servers telling them to download the new file and remove the old one. This ensures remote users can go to their local server and have the latest version of a training module, even if part of the greater JDISS network failed.

Second, the proxy server provides caching. The server detects which Web sites are being accessed most often by its clients. It then stores a copy on the local server so the clients do not have to go to the original site every time the frequently used site is called. Figure 2 illustrates a possible proxy server architecture for JDISS on-line training. (Netscape 1996) The Netscape browser extends the clients ability to interpret information through the use of plug-ins, client based helper applications or delivery of Java applets. This architecture has been copied by the other popular browser in the market, the Netscape Explorer. (Flynn 1996) Finally, the Netscape browser will be incorporated in JDISS in late summer 1996. It is also one of the applications included in MCM.

2. Java and JavaScript

Java is an object-oriented programming language developed by Sun Microsystems for use in programming embedded systems. According to SunSoft (1996),

> Java is a simple, robust, object-oriented,
> platform-independent multi-threaded, dynamic
> general-purpose programming environment. It's
> best for creating applets and applications for the
> Internet, intranets and any other complex,
> distributed network.

It is important to note that first and foremost, Java is an object-oriented programming language, similar to C++. The word processor I used to write this thesis could be written in Java, loaded on my hard drive and executed as a standalone
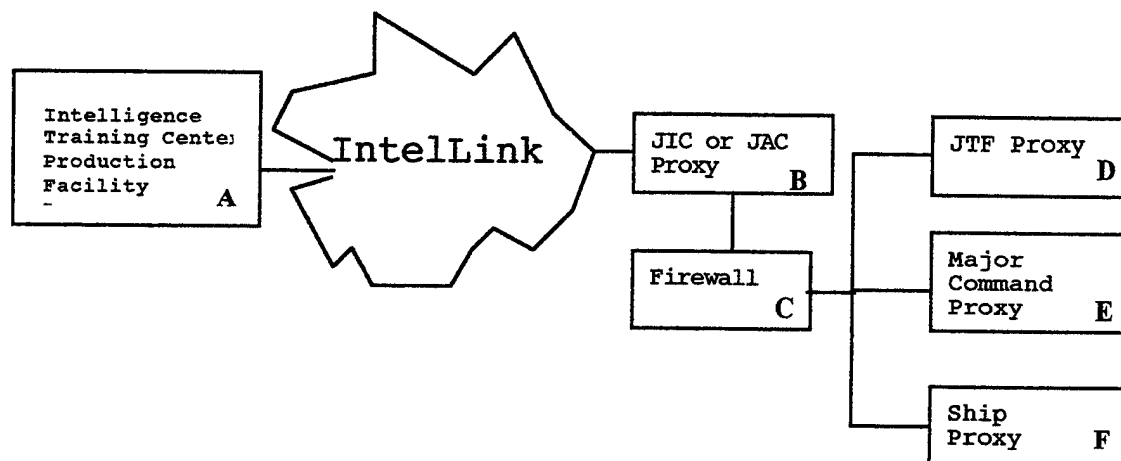
Figure 2. How Proxy Servers Ensure Information Integrity.

Training lessons are produced at the Intelligence Training Facility (A) and stored at a JIC (B). Users in the field (D, E, F) check with their JIC (B) to see if new material is available and download it for local storage. A JIC firewall ensures the field commands do not go out of theater for their training information. If any links between the commands are severed, local users can still receive information from their proxy server.

program. Additional properties of the Java run-time

environment allow for the creation of "safe, dynamic, cross-

platform, active networked applications," or applets.

(Naughton, 1996)

An applet can be thought of as Java ready piece of code

residing on a server and associated with a Web page. One

important difference between a Java applet and a traditional

client/server application is that Java is complied as platform

independent byte code, instead of machine native code. When a

viewer calls up an applet enhanced Web page, the server sends

Java byte code with it. If equipped with a Java ready browser

such as Netscape Navigator 2.0, Microsoft Explorer 2.0 or

Sun's HotJava, the client will dynamically interpret the byte

code.

The interpreter and run-time system are collectively know

as the Java Virtual Machine. (Flanagan, 1996) This scheme

allows Java to be a platform independent language. As long as

a computer has the Virtual Machine, it can run an applet from

any other machine, be it Unix, PC or MAC. Once the applet

finishes its task or the user changes to another URL, the Java

garbage collector removes the byte code from RAM. This new

scheme of delivering executable code via the Web has alarmed

some security managers. Appendices B and C address Java

security concerns and summarize applet capabilities.

Java applets can reside on local or remote servers.

Parameters may be sent to the applet from an HTML page that

42

tells it how to behave. Java is so flexible that two HTML
pages can send differing parameters to the same applet and
receive the same service with modifications reflecting
customized instructions.

JavaScript provides a client side environment to run
small data check or applications in an HTML page. The
functions are provided by a script embedded in the HTML page.
A JavaScript enhanced browser such as Navigator or Explorer
2.0 is required to run a JavaScript. Common uses for
JavaScript include the checking of data parameters before the
information is forwarded to a server and the provision of
interactive enhancements such as radio buttons or text boxes.
In the future JavaScript will be the language for customizing
remote applets located on a server. (Goodman 1996)

3. Migration to CORBA

CORBA defines the way objects, clients and servers within
a distributed computing environment interact. The OMG sets
the standards for the object inter-operability framework. The
consortium includes DoD members. Additionally, the JDISS PMO
has identified CORBA as a future architecture that the system
must support. (OMG 1996a, Tibbitts 1995)

CORBA will be the distributed object architecture of
choice for several reasons. First, CORBA supports the Unix,
Windows, and MAC operating systems while ActiveX only works
with the Windows and Windows NT. Currently, all JDISS nodes
are Unix workstations. Second, the OMG has about a two year

head start on Microsoft. Most ORB producers have released the second version of their product and are working on the third. Finally, numerous companies such as SunSoft, Iona, and Hewlett Packard have released commercial ORB software. Only Microsoft produces ActiveX at this time. This will probably be the case until Microsoft determines whether its developers, or an independent board, will set ActiveX standards. (Gage and Mardesich 1996)

Additionally, Java applets will work in both a non-CORBA client/server migration system and an ORB based system. Several features of Java will ensure any applet based applications can migrate to the CORBA architecture. First, both technologies employ an object oriented view of application design. Second, Sun as a member of the OMG has already implemented a Java IDL. The Java IDL allows Web browsers to run applets. The Java IDL system employs an ORB capable of inter-acting with other ORBs. Sun will also develop an IIOP module which will allow for direct connection of the Java ORB to CORBA 2.0 products. (SunSoft 1996, Linthicum 1996)

The synergy between Sun, OMG, and Netscape ensures that their products and standards will complement each other. (Gage and Mardesich 1996) Since most of the hardware and software base of the JDISS system resides on SUN platforms running the Sun OS or (soon) Solaris, an on-line training system relying on Java enhanced HTML pages read by the

44

Netscape browser will easily migrate to a CORBA backbone. The system will not have to be scrapped during the migration to CORBA. Appendix D describes current JDISS migration to the Defense Information Infrastructure Common Operating Environment (DII COE).

V. PROOF OF CONCEPT: DESIGN AND IMPLEMENTATION CONSIDERATIONS

The on-line proof of concept illustrates how Web based training using Java and JavaScript can be implemented today to produce a system which will easily migrate from client/server to CORBA. COBRA 2.0 compliant products were not used, since the JDISS PMO has not settled on which ORB (if any) it will use. Additionally, there was no funding available to procure this software. All the tools used to produce this demonstration were free.

The goal of the demonstration was twofold. First, to show Java applets, JavaScripts, and advanced HTML techniques in use. Second, to illustrate a computer learning tutorial in a Web environment. The on-line demonstration proves that training can be provided for JDISS in a client/server environment enhanced with the latest in Web and object oriented technologies. As such, portions of this training system will be able to migrate with the JDISS system as the distributed object environment replaces client/server. Finally, this proof of concept will demonstrate the power of Java, JavaScript and specialized HTML commands and their applicability toward JDISS on-line training in the current client/serve environment.

As mentioned earlier, the JDISS PMO has developed an application called the Multimedia Collaboration Manager (MCM).

MCM will be the main JDISS tool this thesis considers for delivery of computer instruction because it allows for the simultaneous use of multimedia information in the form of text, graphics, imagery, audio, and video.

These media can be shared in the context of VTC, white board, HTML pages (through the use of Netscape) or chat session individually or simultaneously. With MCM, JDISS course material may be delivered in a variety of formats. Text and graphics can be delivered via HTML. The video capabilities can be used to extend classroom training beyond the lecture hall. Additionally this application can be used for individual apprentice sessions between teacher and student. The white board can be used in a similar way with teacher and student collaborating on mock graphics of a training module to illustrate which button to use or question to examine. In conclusion, MCM will allow the student to access training using the media which best suits his needs. This will provide him with a customized training application.

A. DESIGN FACTORS

Before editing and programming the various HTML documents, scripts, and byte code, the system was planned based on the precepts of Alessi and Trollip (1985) and Jeiven (1994) to ensure coherent structure and flow of the system from frame to frame. Both provide models and rules for planning an on-line computer learning tutorial.

Alessi and Trollip (1985) provide a simple model which governs the flow of information in a tutorial (Figure 3). In the Web demo, each box in the model can be presented at a different Web site. Proper flow is maintained through hyper-links and browser features such as the forward and back commands or an index in an HTML frame. For example, a Web based review test provides a method to judge response in the on-line Web demonstration. Jeiven provides a comprehensive checklist (Appendix E) for computer learning tutorials.

B. DESIGN OF THE WEB TRAINING APPLICATION

The actual Web demonstration has been placed on-line for view at the following URL: http://vislab-www.nps.navy.mil/wpkopper/demo/welcome.html. Netscape Navigator version 2.0 or higher and a multimedia equipped computer are recommended to observe all capabilities built into the demonstration. Warning messages identifying system shortcomings will appear if your hardware or software can not utilize the advanced features embedded into the demonstration.

1. Frames

All modules within the training demonstration have a common look with several standard features. The JDISS on-line training demonstration starts with an introduction page divided into three frames, a title, an index, and welcoming text (see Figure 4). Frames allow the designer to create a common look throughout a series of Web pages by anchoring certain items, such as a table of contents, to the browser

```
┌──────────┐          ┌──────────┐          ┌──────────┐
│ Intro    │   ───▶   │ Present  │   ───▶   │Question &│
│ Section  │          │ Info     │          │Response  │
└──────────┘          └──────────┘          └──────────┘
                         ▲    ▲                   │
                         │    └──────╮            ▼
┌──────────┐          ┌──────────┐          ┌──────────┐
│ Closing  │   ◀───   │ Judge    │   ◀───   │Feedback or│
│          │          │ Response │          │Remediation│
└──────────┘          └──────────┘          └──────────┘
```
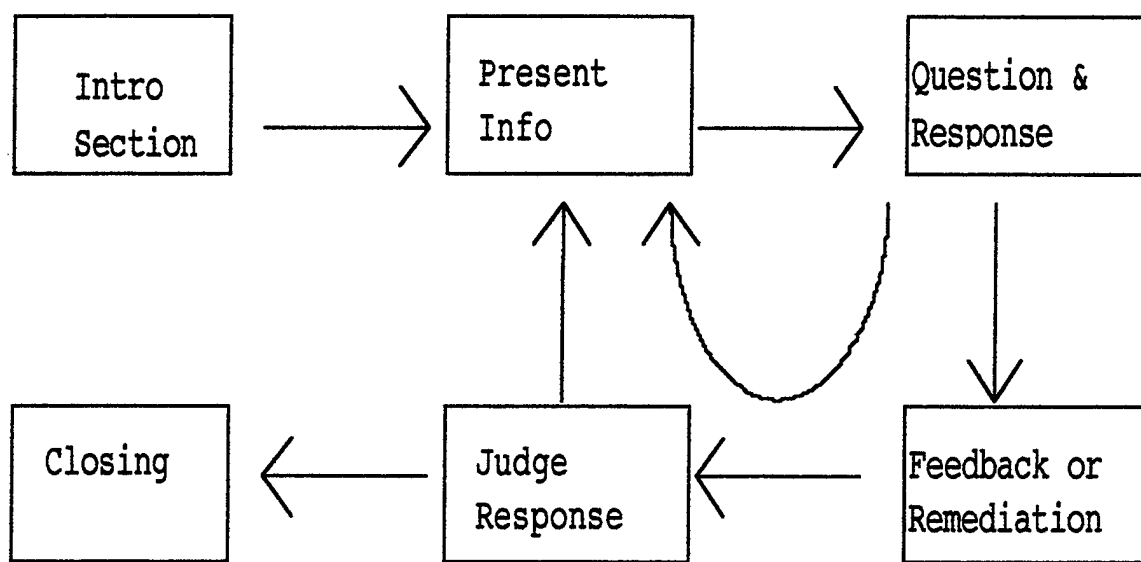
Figure 3. General Structure and Flow of a Tutorial.
After (Allesi and Trollip 1991).
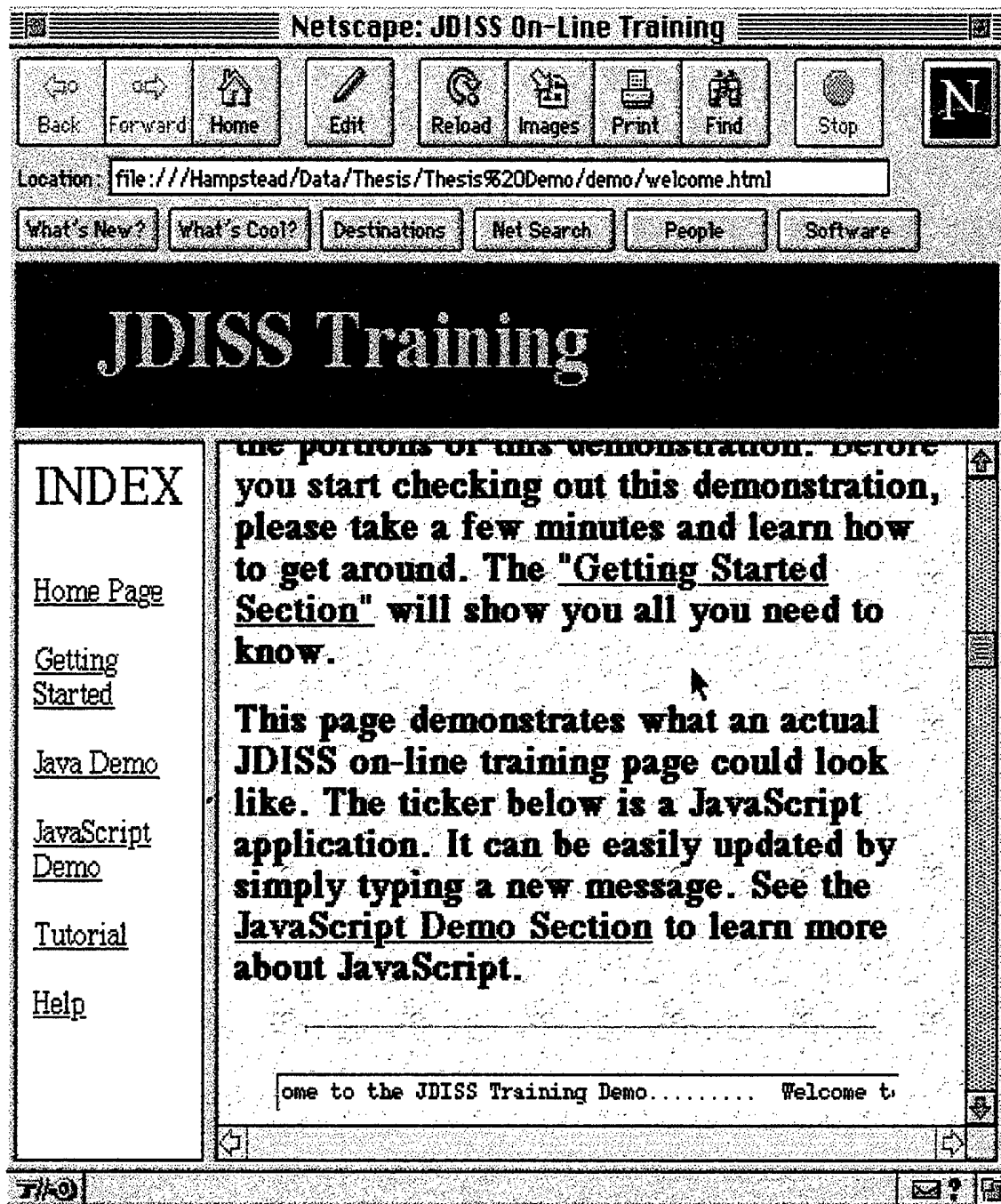
Figure 4. The Welcome Screen.

A Java applet, Blinky, causes the title to flash in different colors, presenting the illumination of animation. Another applet, IntroSound, plays the welcoming message "Welcome to JDISS" while music plays in the background. A JavaScript causes a ticker message, "Welcome to the JDISS Training Demo," to scroll across the box at the bottom of the screen.

window while the user surfs through other pages. In this case, the title and index frames remain constant to provide a uniform location and reference point for navigating through other portions of the demonstration. Appendix F, Section 1, lists the HTML code for the opening frames of the demonstration. Of particular interest are the nested frames within frames which allowed the title frame to extend across the top of the browser window with the index and welcome frames below.

2. JavaScript

The JavaScript demonstration (Figure 5) shows how the scripting language can be used to create a drop down menu with HTML links to other pages. Javascript also serves as the language for preview and review tests within the tutorial. Finally the "Welcome to JDISS ..." ticker in the welcome page relies on JavaScript for its functionality.

Since it is placed in the head of the page, the JavaScript inter-active drop down menu code (Appendix F, Section 2) loads before the HTML text for the page. Thus when the user presses the "access information" button, the request is processed on the client computer for quick return of the applicable URL. Had this function been provided with a Java applet, it would not become available until the applet had been downloaded. This script can be re-used in other HTML documents by changing the names of the buttons and URL
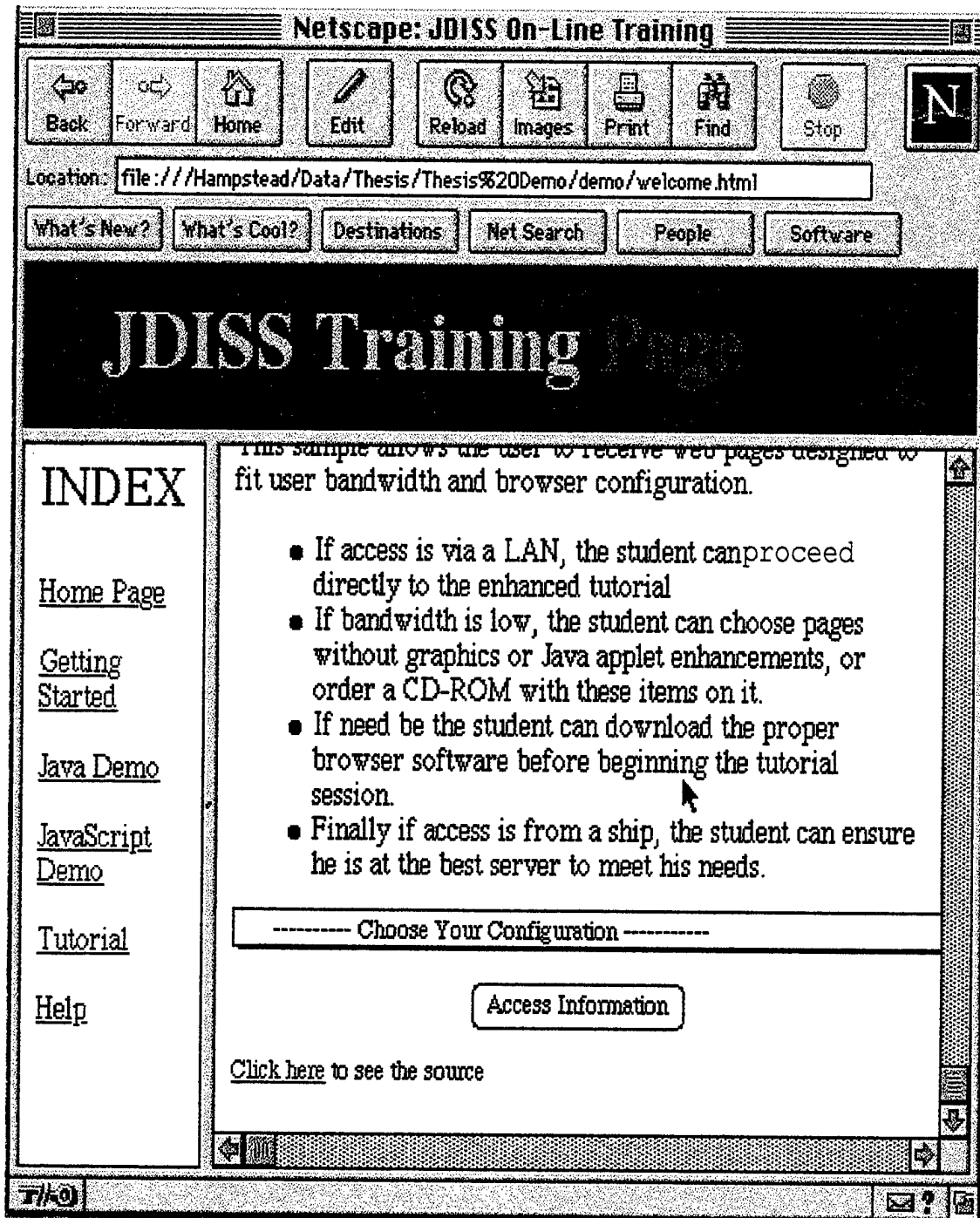
Figure 5. JavaScript Demonstration.

The JavaScript allows the user to choose a configuration and
access HTML pages explaining services available for a given
hardware and software configuration. All computing is done
by the client.

options. Two other scripts, pre-test and review test will be discussed in the tutorial section.

3. Java Applets

The demonstration illustrates the flexibility and potential of Java applets. Java applets appear throughout the demonstration, most often in the Java Demonstration section. Starting with the welcome screen, the flashing "JDISS Training Page" in the top frame is a Java applet, Blinky, which takes parameters from the HTML code for that page to designate the message to flash and the background color to display. Another applet connected to this page, IntroSound, shows how an audio welcome can be added to a Web page. When the user opens this page, he hears an audio "Welcome to JDISS" greeting as well as background music. In both cases, the code for the applets was adapted from published Java demonstrations, modified, and re-compiled to fit the needs of the page. (Lemay and Perkins 1996)

In the Java demonstration section, three capabilities of Java applets are illustrated. First, the applet Rotator (Figure 6) is shown with some parameters which make it stick out from its Web page. This applet takes an image and displays sections of it for a given period of time. Quick rotation of the sections gives the illusion of animation. Appendix F, Section 3, contains the code for Rotator. (NIH 1996) In the first demonstration of Rotator, the size of the display box has been purposely made larger then the image

54
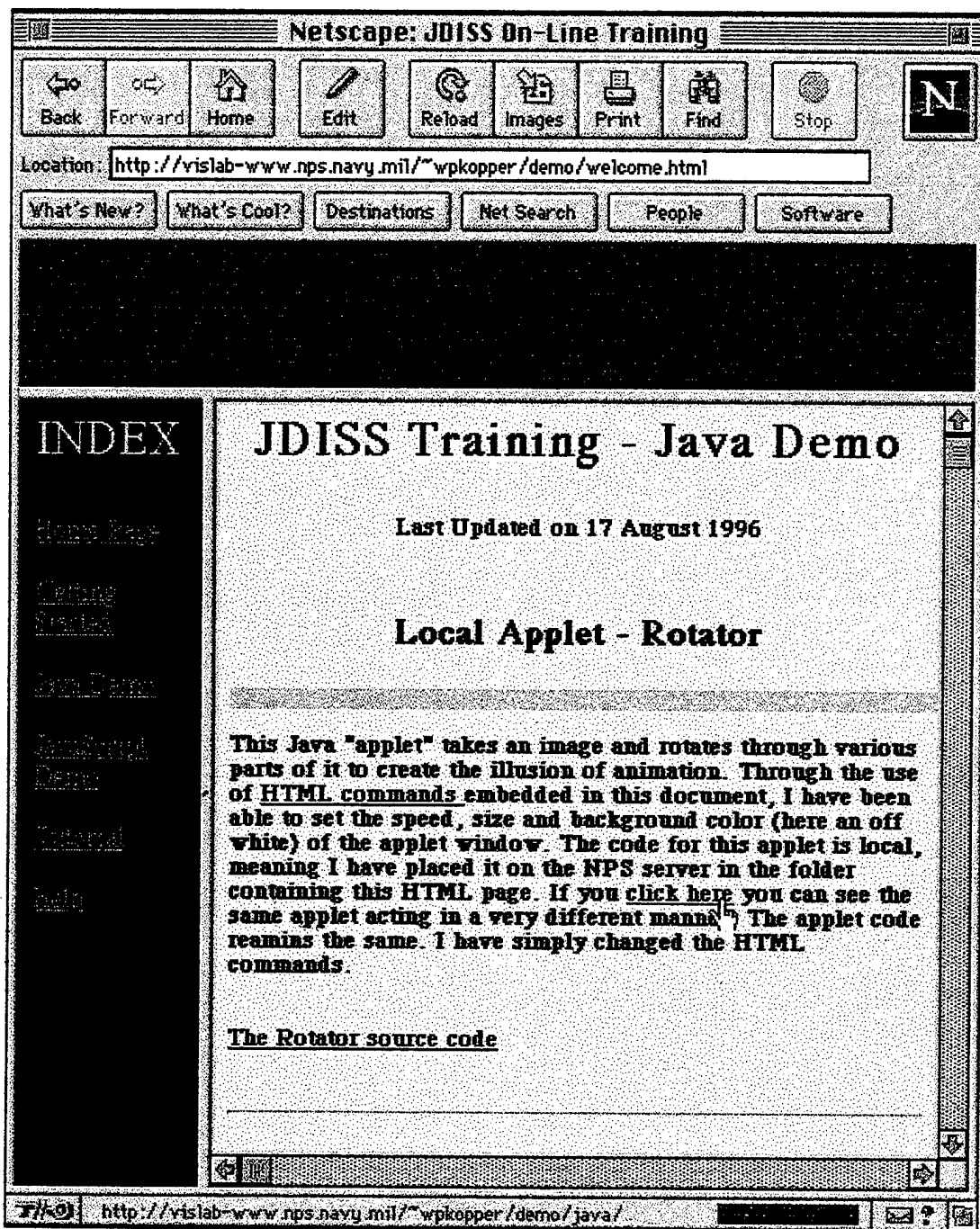
Figure 6. Java Applets Demonstration.

Java Applets, such as Rotator, can be used in many ways. Here, the illusion of a moving line is provided in the gray box.

which is being rotated. In addition, the background color for this applet has been changed to an off-white in order to show the area of the Web page controlled by parameter commands in the HTML code. The parameters are set in the HTML page as shown below:

```
<APPLET = "Rotator.class" width=500 height=10>

//This command calls the applet Rotator and sets up a
500 x 10 pixel display box.

<PARAM name=image value="Line.gif">
<PARAM name=bgcolor value="FFFFCC">
<PARAM name=rate value ="30">

//These parameters designate the image to be rotated,
"Line.gif," the back ground color for the 500 x 10 pixel
box, in this case an off-white, and the rotation rate of
the image in frames per second.

</APPLET>
```

Moving to the next Java applet page (Figure 7), the same compiled applet code is called, yet the effect is much different. This is due soley to changes in the parameters embedded in the HTML code. Using a single image, "animate.gif," it looks as though multiple images are being displayed from the same window. Actually the Rotator applet G takes four areas of 113 by 90 pixels from a single image and
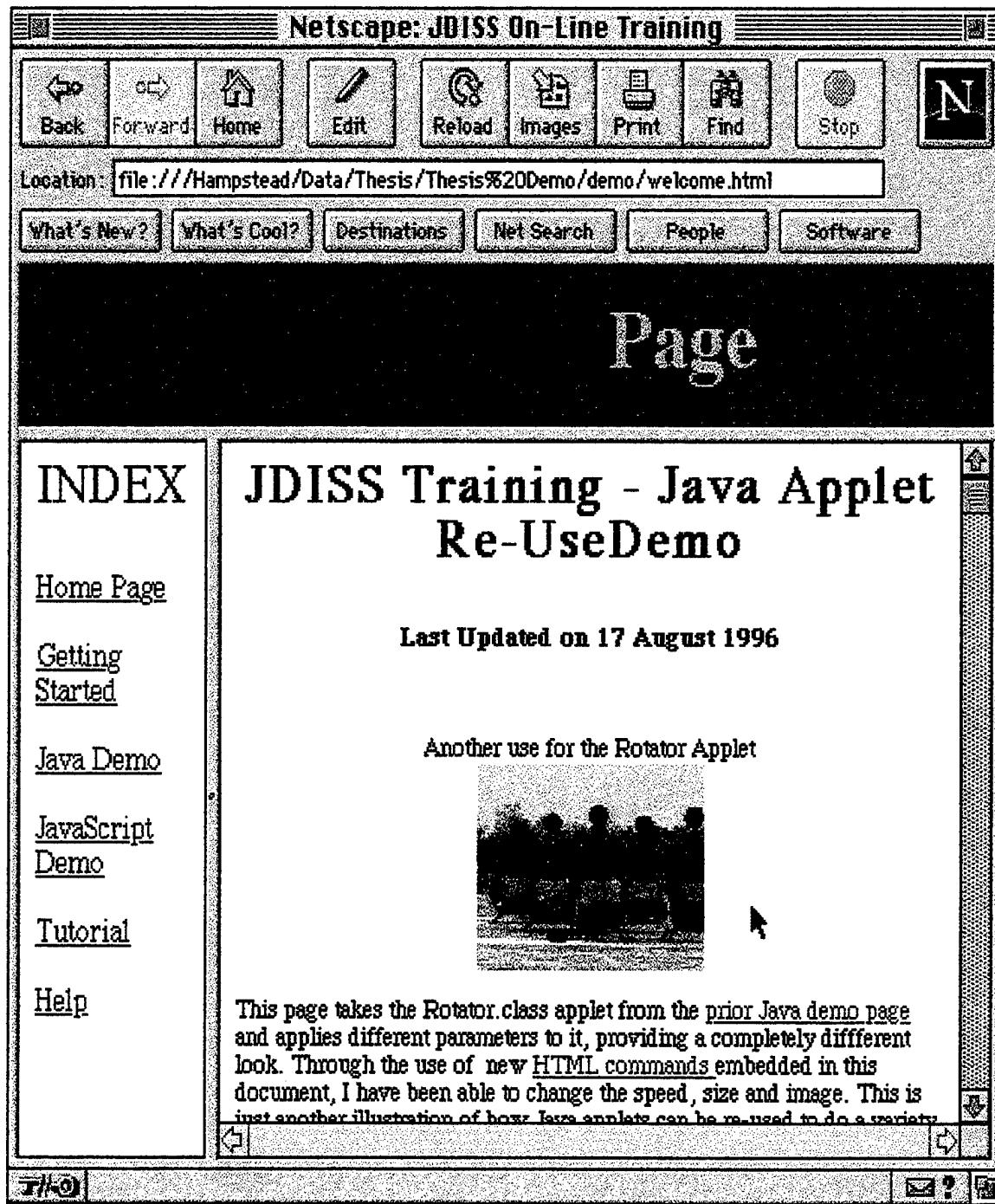
Figure 7. Another use for Rotator.

This time the applet rotates different portions of an image to create a slide show.

displays them in sequence. The parameters are set in the HTML

page as shown below:

```
<APPLET CODE="Rotator.class" width=113 height=90>

//This command calls the same applet code, but
designates a square display window.

<PARAM name=image value="animate.gif">
<PARAM name=rate value="2">

//These parameters call a different image, "animate.gif,"
and display it at a rate of two frames per second.  No
background color designated, red will be displayed if
the  applet is not running.

</APPLET>
```

In addition to using the same applet in two different ways,

the Java demonstration shows how a remote applet, "Bubbles,"

can be called from a remote server and embedded in a Web page.

The applet actually resides on a remote server, yet it can be

embedded in a local page using the following HTML code:

```
<APPLET CODEBASE = "http://java.sun.com/java.sun.com/
applets/applets/Bubbles/" code = "Bubbles.class"
width=500 height=500>

//This command calls a remote applet based on the URL
where it is displayed and the name of the applet.  A
display box of 500 x 500 pixels is designated locally.

</APPLET>
```

The "applet codebase" command allows applets to be compiled

once, placed on a server, and called multiple times for re-use

if proper file access is granted to remote users.

58

4. Tutorial

The tutorial section is actually a demonstration within a demonstration. Starting with the introduction page (Figure 8), three main areas may be accessed, pre-test, review test, and tutorial text. By taking the tutorial, the user can learn an actual JDISS Basic Operator Course (JBOC) lesson. After reviewing the directions and objectives, the student can take a JavaScript pre-test to measure knowledge of the material. When finished, the test can be graded. If any questions are missed, their numbers are reported at the bottom of the browser window and the student can access the correct answer and further information about the topic in the far right frame as shown in Figure 9. The pre-test serves as a way to determine which material should be studied during Alessi and Trollip's (1985) "present info" step.

Following the pre-test and review, the student moves to the lessons. These lessons come directly from the JBOC manual. They have been enhanced in HTML and indexed in a separate frame for easy navigation (see Figure 10). The lessons are the present info section in the Alessi and Trollip (1985) model.

After completing the lessons, the student can take a review test to see how well he has learned the topic. The review test relies on another JavaScript, however this script grades each question as it is answered and records the final

59
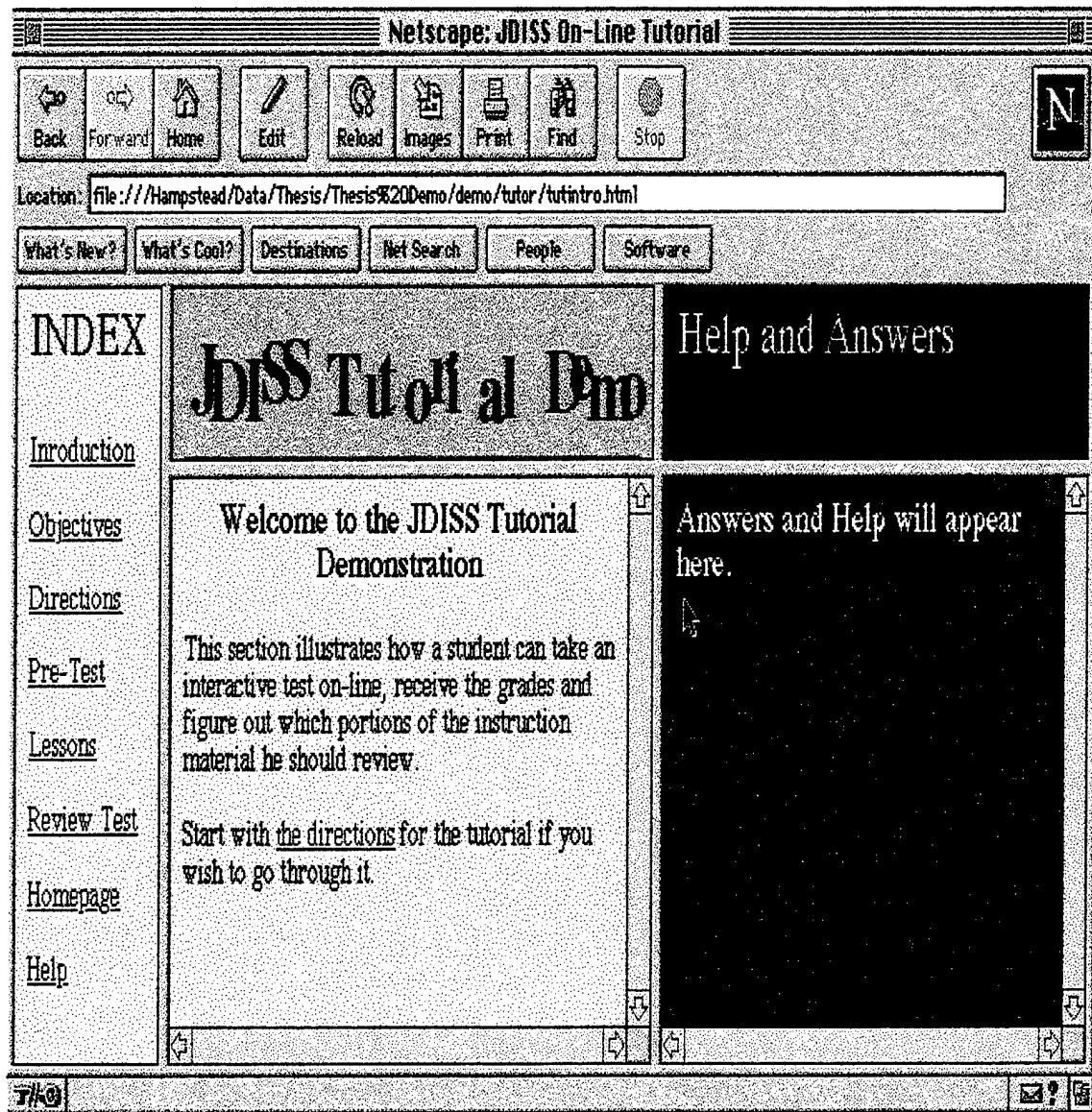
Figure 8. Tutorial.

The beginning of the tutorial provides a new index for easy navigation

Netscape: JDISS On-Line Tutorial

Back | Forward | Home | Edit | Reload | Images | Print | Find | Stop

Location: http://vislab-www.nps.navy.mil/~wpkopper/demo/tutor/tutintro.html

What's New? | What's Cool? | Destinations | Net Search | People | Software

INDEX

Help and Answers

JDISS Tutorial Demo

System

Question #2: "Smart push" compliments "pull" architecture:

○ A). True
● B). False

Question #3: JDISS has been deployed in the following exercises *except*:

● A). Desert Storm
○ B). Provide Promise
○ C). Restore Hope.

Grade Test | Reset

Correct Answers

#1, #2, #3

II. Automated Dissemination

A. Pull architecture is a systematic approach designed to take advantage of emerging data transfer technologies while easing the communications burden on traditional intelligence dissemination systems. Pull architecture allows intelligence consumers to remotely access desired data at the time and place of their choosing as opposed to the intelligence producers "pushing" large amounts of information indiscriminately. A thorough knowledge of Theater and National level intelligence products and producers is essential to maximizing the benefits of Pull architecture.

Time to study! You missed question(s) # 1 2

Figure 9.   Pre-test.

The Pre-test provides a grading mechanism (page bottom) and hyper-links to review material when a question is missed. The student can find which questions he got wrong by clicking on the "Grade Test" button. The numbers of wrong questions appear in the bottom bar. Hyper-links in the "Correct Answers" section cause review material to be displayed in the right hand frame.

Location: file:///Hampstead/Data/Thesis/Thesis%20Demo/demo/tutor/lessonfr.html

What's New? | What's Cool? | Destinations | Net Search | People | Software

INDEX

What is JDISS?

Automated Dissemination

JDISS Deployed

JDISS Migration

JDISS Communicatio

Conclusion

Review Test

Tutorial Intro

Homepage

Help

Lessons

II. Automated Dissemination

A. Pull architecture is a systematic approach designed to take advantage of emerging data transfer technologies while easing the communications burden on traditional intelligence dissemination systems. Pull architecture allows intelligence consumers to remotely access desired data at the time and place of their choosing as opposed to the intelligence producers "pushing" large amounts of information indiscriminately. A thorough knowledge of Theater and National level intelligence products and producers is essential to maximizing the benefits of Pull architecture.

B. Complementing Pull architecture is the use of "Smart Push" techniques. Smart Push is designed to more effectively use traditional intelligence dissemination methods and downward directed intelligence. Smart Push is based on the understanding that not all intelligence consumers will have access to the pull architecture and also that situations will

Figure 10.   Lessons.

After taking the Pre-test, lessons hyper-linked to the index may be reviewed.

score of the test (see Figures 11 and 12) which could then be forwarded to a central grading authority for review.

Appendix F, Sections 4 and 5, illustrates the JavaScripts for these tests. Both were found on the Internet and modified for the on-line tutorial demonstration.

Figure 11. Review Test.

The Review Test keeps a running tally of correct answers.
Each question appears on a different screen Once an answer
has been entered, the student can not change it.

Figure 12.   Review Test Grade Screen.

Following selection of answers, the Review Test is graded and
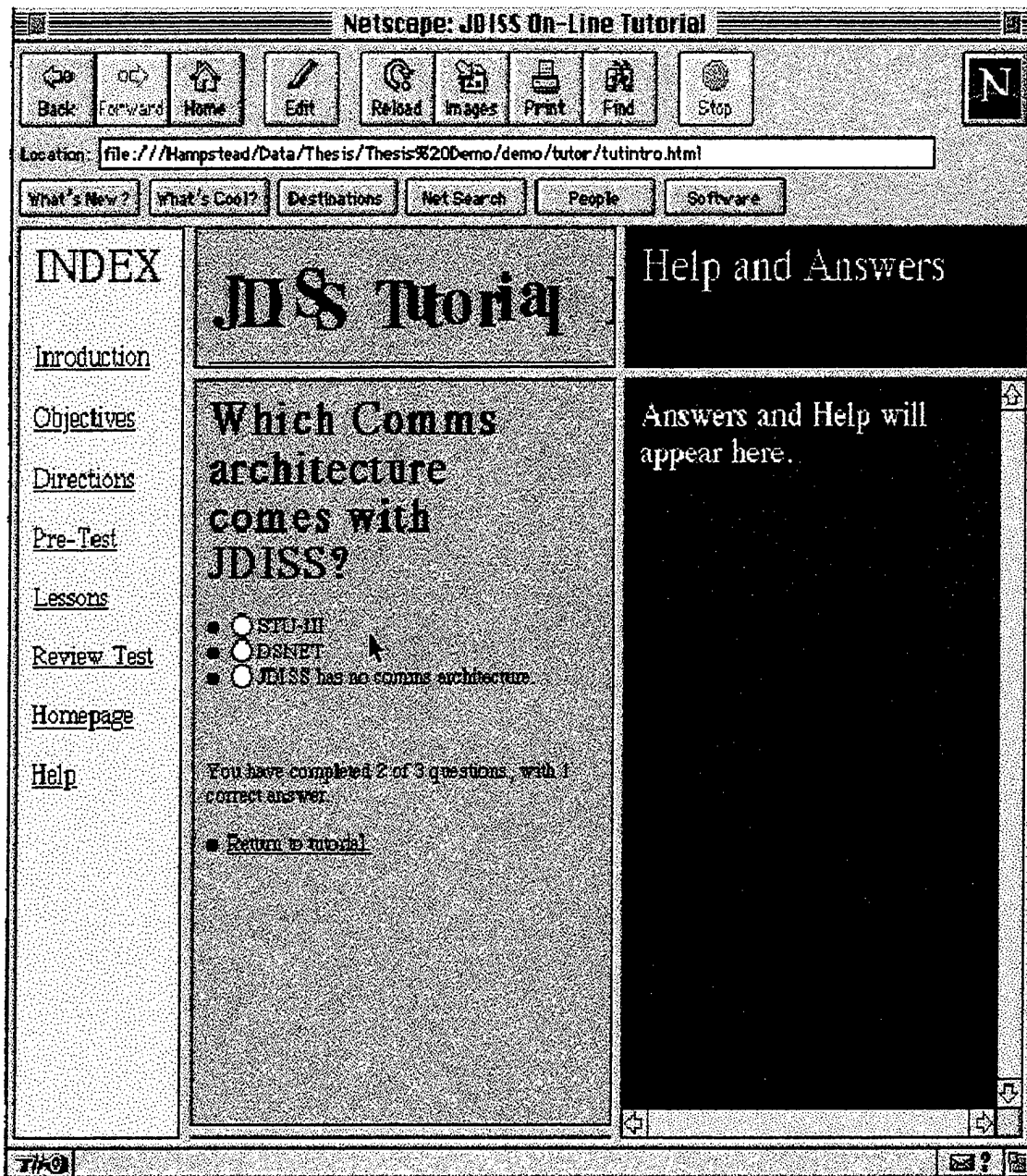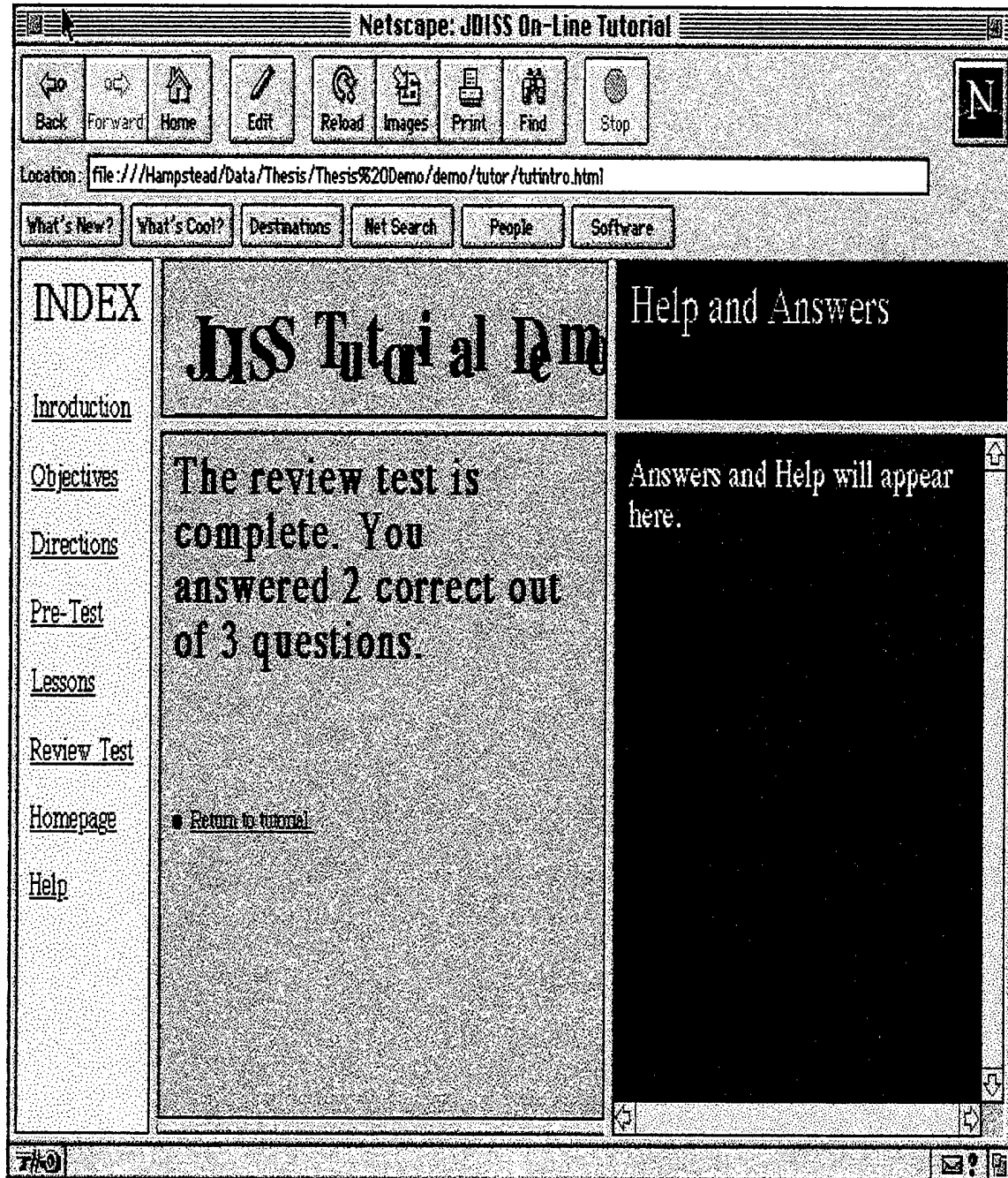a tamper proof score is provided.

VI. LESSONS LEARNED

A. TOOLS

A variety of applications were used to develop the demonstration. Some worked well, while others were sent back to the manufacturers. All products with the exception of the text editor were version 1 or beta products. In general these tools produced useable HTML documents, JavaScripts or Java applets. However, they are often hard to use and in many cases simply did not produce what was expected.

The tools break into two categories: HTML editors, which were used to produce Web pages with embedded Java parameters and JavaScripts, and Java applet development environments, used to write code and compile Java applets.

1. HTML Editors

The ultimate HTML application would provide true WYSIWYG editing for HTML text and visual GUI tools for embedding Java and JavaScripts. Unfortunately current tools fall somewhat short of this expectation. The on-line demonstration employed three editors to create Web pages, Netscape Gold, Corel Word Perfect and Apple Simple Text. Often all three were required to create a Java or JavaScript enhanced Web page.

The Web editor enhanced Netscape Navigator Gold is the easiest of the editors to use. Due to the constantly evolving nature of the Netscape browser, three beta version of the

67

Navigator Gold, 3.0b2, 3.0b5 and 3.0b6, were used in the development of the Web demonstration pages. Each new iteration surpassed its prior release. Netscape Gold allows any HTML page to be opened in the editor, be it on the local machine or the Web. In editing mode the user can easily manipulate text size, type face, and color through the use of options located on a palate. Tables and URL links can also be created. One draw back was the lack of a color palette. When changing the color of the text, the page author is given a color wheel and asked to choose percentages of green, blue, and red which make up the selected color. A palette of 36 or 64 colors would have been much easier to use. The lack of a comprehensive visual environment for incorporating advanced HTML commands ensured that Gold was used only to manipulate text and background for the pages. It is unfortunate that Netscape, often the creator of new HTML commands does not incorporate them into their development application.

Word Perfect fared worse then Gold. It is good for converting a document written in the application into HTML, but poor for creating an HTML document from scratch. This is probably due to the age of the product. It was developed in 1995. The HTML portion of the application relies on the Navigator browser to illustrate what the finished file looks like on the Web. Often the results were surprisingly

different from what was expected. The application had problems converting centering commands and font sizes into HTML.

Regardless of the HTML editor used, the final page was often completed in a basic text editor, Simple Text. This application was used to incorporate complex commands such as those for frames, applets, and JavaScripts by manually typing in appropriate HTML tags. These files were saved and opened with the Navigator browser to ensure they met expectations. In conclusion, a comprehensive knowledge of HTML code and tags is necessary to publish all but the most elementary of Web pages.

2. Java Development Environments

Two applications were used to develop Java applets, SunSoft's Java Development Kit (JDK) 1.02 and Symantec's Cafe. Both are first attempts at a Java development tool.

The JDK can be downloaded for free form SunSoft. It includes a compiler, class library and run time for standalone applications and an applet viewer. The development environment is non-existent. Applets and applications are developed in a text editor, based on the developers knowledge of Java. This can be gained from the extensive Java documentation and sample code included in the JDK. The compiler works well enough, but flashes constantly changing percentages of progress, even though no code is being compiled. The applet viewer works well, but is unnecessary if

69

one has a Java enhanced Web browser to view the applet. The
application runner is a must if you want to develop standalone
applications in Java.

Cafe provides a visual development environment for
creating Java applets. Unfortunately, Cafe falls short of all
promises. In fact a recent review called this commercial
product "a pre-release version." (Crabb 1996) The version
used to develop the on-line demonstration was so full of bugs
that it was returned to Symantec.

In conclusion, a product which allows for rapid Java
applet development akin to Borland's Delphi is still one or
two versions down the road. The Web proof of concept would
not have been possible without the use of multiple books
explaining HTML, JavaScript and Java.

B.   DEVELOPMENT PROCESS

The first step in designing a Web training site is to map
it out to ensure that links referenced in the site will have a
corresponding HTML page, graphic, or applet. Another
important aspect of developing an on-line site is keeping
track of the which files will be called by a given URL. Many
pages will call on the same Java applets or show up in the
same frame environment. Mapping out the storage hierarchy for
the site makes file management much easier.

When using frames, Java, and JavaScript, it is important
to remember that not all Web surfers will have a browser which
can understand these features. Knowledge and use of the

70

warning commands which allow the user to understand why he can not access the information with his browser will avoid future complaints that the site does not work.

Finally, the use of the browser during the development process is a must. Frequent views of the file with the browser show the author what the user will actually see. Mistakes and errors can be caught before the page is loaded on the server for use by the greater Web community.

VII. CONCLUSIONS

This thesis explored how JDISS training material could be placed on the Web in a manner such that these modules would remain compatible with future distributed object technology. It suggests that HTML, Java, and JavaScript will be the primary development languages of the Web. Additionally, it proved that these tools can be used to develop an interactive Web training site incorporating JDISS tutorials currently available in published tests or static HTML pages. With the rapid advancement and coming merger of Web and distributed object technology, numerous areas of research will have to be explored to keep a Web training site up-to-date.

Computer learning applications have traditionally been standalone applications. The emergence of the Web and the coming of distributed object technology will radically change this model. Customized training applications created with Java and JavaScript objects on an HTML framework will allow for the creation of a customized application, built dynamically, for the user. The architecture for this new environment will be the next generation of the Web with CORBA compliant ORBs exchanging information via the Internet Inter-Orb Protocol (IIOP), the probable replacement for HTTP. While the ORB and its object services are not yet readily available for implementation, it is clear that an understanding of Java,

73

JavaScript, and HTML will help developers to easily migrate to the next architecture.

A. LIMITATIONS

Although the demonstration tutorial utilized advanced HTML, Java and JavaScript on the Web, the interface is still relatively static. The user can determine and utilize only that material necessary for him to achieve the course objective, but he still cannot develop a customized training module on the fly.

Additionally, this thesis, while predicting the advent of the Object Request Broker, did not prove the viability of using an ORB. Commercial CORBA 2.0 compliant ORB's are available from several commercial vendors, but object families tailored for training are not yet available.

Finally, the use of SGML was precluded due to the lack of tools to support detailed tagging and searches on the Web. These functions require an SGML enhanced browser application.

B. AREAS FOR FUTURE RESEARCH

As the Web evolves from a static arena of fixed sites to the tool for developing a tailored site with distributed objects, numerous new areas will have to be explored. Most of these areas focus on new or future enhancements which will be provided by CORBA.

1. Migration to Distributed Object Architecture

Applets written in Java can be placed in an IDL wrapper for use in a distributed object environment, yet so can

74

applications written in other languages. Additional research into migration to the CORBA standard would provide an idea as to which types of applications could be broken into IDL wrapped objects for use in a distributed environment.

2. Creation of Customized Training Modules

Once objects are capable of combining over a network, a method to manage them must be adopted in order to insure the right object or iteration of that object is provided when called. Further research into the implementation repository standards will identify how to ensure the right object is called during dynamic invocation. Another type of information to be managed is text. A study of SGML tags to see whether they will allow formatted documents to be parcelled up and re-combined into customized training documents is recommended.

Java interfaces to data bases may be part of the solution to managing the customized training module or network application. An examination of Java front ends, data base applications and their capabilities could provide the tools which would allow for creation of customized documents and applications, based on a user's query. These items will probably take the form of a custom Web page.

Finally, the method for querying the Web to create the tailored service should be examined. Decision Support System (DSS) methodologies could be the basis for the users describing what he wants to do. A DSS query to a data base or

implementation repository may be the way to ensure the user

gets the proper customized document or application.

APPENDIX A. JDISS STANDARDS AND GOVERNING BODIES

A myriad of boards and regulations dictate JDISS configuration and standards. New proposals for JDISS services must adhere to these criteria or risk the possibility of rejection. An understanding of these multiple boards, their relationships, and the standards they decree will ensures that the proposed JDISS Training Architecture will operate within the proscribed DoD environment.

Although JDISS is a joint program, DoDIIS has delegated day to day management of the program to the Office of Naval Intelligence (ONI). ONI manages JDISS configuration through the JDISS PMO, a division the Systems Directorate (ONI-7). This by no means allows the Navy to dictate JDISS standards for the other services. JDISS service requirements are developed by the JDISS Configuration Control Board (CCB) a body subservient to the DoDIIS Management Board (DMB) which acts as the DoDIIS CCB. As such the JDISS CCB must ensure all recommended hardware and software configurations comply with DoDIIS architectures and standards. Furthermore all suggested JDISS hardware and software configurations must be compatible with other DoDIIS systems. The JDISS CCB draws its membership from the JDISS PMO, unified commands, Defense Intelligence Agency (DIA), national agencies, and the service support staffs. Two additional boards, the JDISS Engineering Review

Board (ERB) and JDISS Integration and Test Laboratory (ITL) advise the JDISS CCB on technical and financial matters. (JDISS CCB 1994)

A group of joint and service training councils holds responsibility for JDISS training standards. At the DoD level the General Intelligence Council (GITC) provides policy for all intelligence related training. Current GITC guidance emphasizes JDISS training as a common skill for all intelligence professionals. Furthermore the GITC identifies computer learning and multimedia as areas for delivery of future training. (GITC 1994)

The JDISS PMO has designated the Joint Intelligence Training Activity Pacific (JITAP) as the Core Curriculum Manager for courseware development. JITAP operates under the cognizance of both GITC and JDISS PMO to ensure all JDISS training materials comply with various General Intelligence training System (GITS) standards.

APPENDIX B.   JAVA SECURITY

The state of the Web technology continues to rapidly advance. In two years the Web has evolved from a collection of static pages written in Hyper Text Markup Language (HTML) to a myriad of multimedia enhanced, swiftly changing Web sites. The latest Web advancement has been the introduction of sites supported by executable programs or applets, written in SunSoft's Java language, which travel with the page to provide interactive or multimedia enhancements.

Since the Web allows a client to view a nearly unlimited amount of home pages, delivery of unknown executable code from random sites has raised numerous security concerns. The client has no way of knowing what applets, if any, will be delivered when he chooses a Universal Resource Locator (URL) to receive a Web page. This immediately raises concerns that malicious code or a programmed threat, e.g., a trojan horse, could enter the client machine and corrupt or compromise the users files and data. (Russel and Gangemi, 1991) While any good security manager must be concerned with the threats associated with Java applets, the introduction of Java enhanced Web pages on secure government networks does not increase the likelihood of a malicious attack provided the network itself remains secure.

Java changes the security model. The security manager can no longer limit his scope to executables on LAN clients

79

and servers. With Java, executable code can arrive form anywhere on the Web. A user surfing the Web could surreptitiously download a malicious applet capable of damaging the system and/or data. Security procedures must guard against attacks from anywhere users connect. According to Bank (1996), there are four major types of attacks which the system must protect against:

- Integrity Attacks - such as deletion/modification of files or killing processes or threads.
- Availability Attacks - filling all memory or the file system, creating thousands of windows or otherwise keeping the machine from being used.
- Disclosure Attacks - mailing information about your machine such as password or sensitive files to another network or adversary.
- Annoyance attacks - such as causing the machine to emit strange sounds or display obscene picture on the screen.

Programs that cause the above types of damage may appear to be legitimate applets, yet upon arrival they initiate an attack against the machine and its resources. Knowing of these concerns, the Java developers designed numerous security checks into the language and virtual machine.

Java has a variety of features which make it difficult to introduce a nasty applet from a server to a client. The

language provides protection at three layers within the Java
environment:

- The Java language itself

- The standard set of Java libraries

- The Java Web browser

Within the language, Java specifies entry control for
variables and methods, disallows pointers, and provides
garbage collection. These features allow the programmer to
control access to objects and memory, for example a file
object, to ensure the class can not be changed by the applet.
In the case of a file, the applet would only be able to read
it. The lack of pointers ensures that an applet can not have
an instruction that points to an outside bit of harmful code.
Garbage collection clears memory that is no longer being used.
Thus, a covert instruction can not remain in RAM. (Banks
1996)

The first security check comes when the programmer
compiles the applet into byte code. The compiler will raise
exceptions to security violations. Next, during
interpretation on the client side, the sub-classes found in
the applet are checked against the local Virtual Machine class
libraries to ensure they inherit methods from the super-
classes on the client. When an applet brings a new sub-class
which is not part of the class library on the client machine,
that sub-class is isolated and checked by the ClassLoader,
another part of the run-time, for Java compliance. Methods

that create new or altered super-classes are prohibited.

As the virtual machine interprets the byte code it checks for adherence to the security rules. In addition to checks by the ClassLoader, a Java SecurityManager object checks the methods of other objects during run-time to ensure they do not violate the Java access rules. (Banks 1996, Sunsoft 1996) The lack of comprehensive documentation for the SecurityManager may be the biggest stumbling block in Java security certification.

Finally the Java browser provide additional protection features. The browser can be configured for any of four protection realms:

- unrestricted - enables from anywhere applets to do anything.

- firewall - enables applets within the firewall to do anything.

- source - enables applets to do anything only within their origin [Internet] host.

- local - disenables all file and network access.

The compiler tags an applet with its origin. The run-time environment checks the tags of each applet for its origin and can dynamically restrict execution of any code from a prohibited realm. A future Java feature will be the addition of a digital tag attached to the applet when it is compiled into byte code. This tag, based on public key encryption, will identify who originated the code and *guarantee its*

integrity. (emphasis added) Appendix D, exhibits applet access capabilities based on browser environment. (Lemay et,al 1996, Sunsoft 1996)

Although Java is still in the documentation and certification process, placing applets on secure networks should be safe as long as a few security rules are followed. First, applets should only be loaded into designated directories on specific servers. Access to these directories should be protected using standard Unix directory access permissions. Since an applet must have the suffix ".class" at the end of its name, non-designated directories can be searched and for surreptitious applets. Additionally, all browsers should be set to the applets the firewall realm.

Of course the browser is only as secure as its source. It is imperative that browsers come verified companies or contractors. Downloading browsers from the Web and placing them upon classified networks should be forbidden. By loading only certified Java browsers, the security manger ensures that applets created by a custom compiler, built for the specific purpose of creating malicious Java code, will be rejected. Verified browsers will only recognize byte code built with the SUN compiler. Spoofing browsers could be designed to recognize applets from a non-authorized compiler, causing them to execute and cause havoc.

With these procedures in place, Java applets become just as safe as any other executable code on the network. Given

Java's built in security, they are probably safer. The introduction of a "trojan" applet would mean either intentional placement on the network by a cleared system user or compromise of the network's firewall. Either violation would not be caused by Java flaws, but by shortcomings in network security. Once the language is completely documented, restriction on the locations of applets may be lifted and they can be placed anywhere on the network.

APPENDIX C.   APPLET CAPABILITIES

Key:

NN: Netscape Navigator 2.x, loading applets over the Net
NL: Netscape Navigator 2.x, loading applets from the Local
           file system
AN: Appletviewer, JDK 1.x, loading applets over the Net
AL: Appletviewer, JDK 1.x, loading applets from the Local
           file system
JS: Java Standalone applications

Stricter ------------------------> **Less strict**

	NN	NL	AN	AL	JS
read file in /home/me,	no	no	no	yes	yes
write file in /tmp,	no	no	no	yes	yes
get file info,	no	no	no	yes	yes
delete file, using File.delete()	no	no	no	no	yes
delete file, using exec /usr/bin/rm	no	no	no	yes	yes
read the user.name property	no	yes	no	yes	yes
connect to port on client	no	yes	no	yes	yes
connect to port on 3rd host	no	yes	no	yes	yes
load library	no	yes	no	yes	yes
exit	no	no	no	yes	yes
create a popup window without a warning	no	yes	no	yes	yes

APPENDIX D.   CSE SS TO DII-DOE MIGRATION

In addition to guidance provided by the various
intelligence configuration boards, broader trends within DoD
effect the development of JDISS architecture. The biggest
trend is the DoDIIS wide migration from Client Service
Environment System Services (CSE-SS) to the Defense
Intelligence Infrastructure Common Operating Environment
(DII-COE). This migration will govern the evolution of JDISS
from its current configuration of tightly integrated hardware
and software suite to a series of portable segments.
According to Essex (1996), "the DII COE is not a system; it
is a foundation for building an open system."

The DII COE consists of three components: the Common
Desktop Environment (CDE), System Administration Services,
and client/server segments. Of most importance to the design
of future JDISS services is the migration of the current
JDISS services to a set of DII-COE segments. Since JDISS
will soon incorporate the Netscape Web browser as a
replacement for Mosaic, the proposed training architecture
must follow the prescriptions of the browser segment.

DII will allow segments from various DoD systems to
collaborate in a common operating environment. It will be
the glue that binds disparate systems and encompasses
commercial architectures, such as the Common Object Request
Brokerage Architecture (CORBA) and Distributed Computing

87

Environment (DCE), into an integrated set of cooperating modules. The benefits of this scheme will allow for a standard common look and feel throughout platform independent environment. (Mitre 1996, Essex 1996)

APPENDIX E.   COMPUTER LEARNING CHECKLIST

- COURSE TITLE SCREEN

- WELCOMING SCREEN

- COURSE MENU

- DIRECTIONS HOW TO USE THE CBT SYSTEM

- COURSE OBJECTIVES

- COURSE PREREQUISITES

- DURATION OF THE COURSE

- REFERENCES TO SUPPORTING MATERIALS SUCH AS BOOKS

- AN "ESCAPE" OPTION TO HELP THE USER EXIT FROM A SCREEN

- COMPLETION STATUS OF EACH TOPIC

- BACKWARD PAGING TO GIVE MORE CONTROL OVER THE LEARNING PROCESS

- INTRODUCTORY SCREEN CONTAINING THE TOPIC NAME AND BRIEF DESCRIPTION OF CONTENT

- LIST OF OBJECTIVES

(Jeiven 1994)

APPENDIX F. SELECTED HTML, JAVA AND JAVASCRIPT CODE

1. HTML for Frames

```
<HTML>
<HEAD>
<TITLE>JDISS On-Line Training</TITLE>
</HEAD>

<BODY>

  <FRAMESET ROWS="75,*">
     <FRAME NORESIZE NAME="top" scrolling = "no"
     src="jdisstop.html">
        <FRAMESET COLS="100,*">
             <FRAME NAME="left" scrolling = "no"
             src="index.html">
             <FRAME NAME="right" scrolling = "yes"
             src="hi.html">
        </FRAMESET>

<NOFRAMES>

     Sorry, this document can be viewed only with Navigator
     2.0 or Explorer 2.0 or later versions.  You can see the
     first document in this set by <a href =
     "intro.html">clicking here</a>.  Thank you.

</NOFRAMES>

</BODY>
</HTML>
```

2. Drop Down Menu JavaScript

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript">

<!-- Hide the script from old browsers --

/* Michael P. Scholtis (mpscho@planetx.bloomu.edu)
   All rights reserved.  February 17, 1996
   You may use this JavaScript example as you see fit, as
   long as the information within this comment above is
   included in your script.

*/


function surfto(form) {

     var myindex=form.dest.selectedIndex

     window.open(form.dest.options[myindex].value, "main",
"toolbar=no,scrollbars=yes");
}

// --End Hiding Here -->
</SCRIPT>
</HEAD>

<BODY>
<FORM NAME="myform">

<SELECT NAME="dest" SIZE=1>

<OPTION SELECTED VALUE="">---------- Choose Your
Configuration -----------

<P>

<OPTION VALUE="http://vislab-www.nps.navy.mil /~wpkopper/
demo/javascript/url.html#lan">LAN access with Netscape 2.0

<OPTION VALUE="http://vislab-www.nps.navy.mil/~wpkopper/
demo/javascript/url.html#dial">Dial in Access with Netscape
2.0
```

```
<OPTION VALUE="http://vislab-www.nps.navy.mil/~wpkopper/
demo/javascript/url.html#browser">Download Netscape

<OPTION VALUE="http://vislab-www.nps.navy.mil/~wpkopper/
demo/javascript/url.html#sea">From
the Sea

</SELECT>
</P>

<P>
<INPUT TYPE="BUTTON" VALUE="Access Information"
onClick="surfto(this.form)">
</FORM>
</P>

</BODY>
</HTML>
```

3. Java Code - Rotator.class

```
/* Rotator - an applet that animates a sequence of frames
   stored as a horizontal strip in a single image.
   http://rsb.info.nih.gov/nih-image/Java/Rotator/

Parameters:
   image   Name of image file, no default
   rate    Rate in frames per second, default = "6"
   bgcolor Background color, default = "ffffff" (white)

This example animates 10 55x68 frames stored in a single
550x68 gif file.

   <applet code="Rotator.class" width=55 height=68>
   <param name=image value="Duke.gif">
   <param name=rate value="4">
   </applet>
*/

import java.awt.* ;

public class Rotator extends java.applet.Applet implements
Runnable {

   int    frameWidth;
   int    delay = 167;   // milliseconds (6 frames/sec)
   int    frame = 0;
   int    bgcolor = 0xffffff; //white
   boolean suspended = false;
   Thread  runThread;
   Image   img;

   public void init() {
       String p = getParameter("width");
       frameWidth = Integer.parseInt(p);
       p = getParameter("rate");
       if (p != null)
           delay = 1000 / Integer.parseInt(p);
       if (delay < 10)
            delay = 10;
       p = getParameter("bgcolor");
       if (p != null)
           bgcolor = Integer.parseInt(p, 16);
       setBackground(new Color(bgcolor));
       img = getImage(getCodeBase(), getParameter("image"));
   }
```

```
public void start() {
    if (runThread == null) {
        runThread = new Thread (this);
        runThread.start();
    }
}

public void run() {
    while (runThread != null) {
        repaint();
        try {runThread.sleep(delay);}
        catch (InterruptedException e) { }
    }
}

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    g.drawImage(img, -frame * frameWidth, 0, null);
    if (++frame  == img.getWidth(this) / frameWidth)
        frame = 0;
}

public boolean mouseDown(Event evt, int x, int y) {
    if (suspended)
        runThread.resume();
    else
        runThread.suspend();
    suspended = !suspended;
    return true;
}

public void stop() {
    if (runThread != null) {
        runThread.stop();
        runThread = null;
    }
}

}// Rotator
```

(NIH 1996)

4.   Pre-test JavaScript

```
<HTML>
<HEAD>
<TITLE>Tutorial, Pre-test</TITLE>

<SCRIPT>

<!-- hide this script tag's contents from old browsers

answer0="B"

answer1="A"

answer2="A"


function scoretest(form)

{

      a=0

      incorrect=""

      if(form.Q1.value!=null&& form.Q1.value!=answer0){

      a=a+1

      incorrect=incorrect+"1 "

      }

      if(form.Q2.value!=null&& form.Q2.value!=answer1){

      a=a+1

      incorrect=incorrect+"2 "

      }

      if(form.Q3.value!=null&& form.Q3.value!=answer2){
      a=a+1
      incorrect=incorrect+"3 "
      }
      if (a==0)
```

```
        {window.status="Congratulations, you are a JDISS pro!
You didn't miss any of the questions answered";}

        else

        {window.status ="Time   to   study!     You   missed
question(s) #   " + incorrect+",";

        }

}



<!-- done hiding from old browsers -->

</SCRIPT>
</HEAD>

<BODY   TEXT="#000000"   BGCOLOR="#FFF0F0"   LINK="#FF0000"
VLINK="#800080" ALINK="#0000FF" BACKGROUND="bac.gif">


<H2><FONT COLOR="#0000FF">Pre-Test</FONT></H2>

<P>
<A NAME="d1"></A>
<FONT SIZE=+1>Question #1: JDISS stands for:</FONT>
</P>

<OL>

<P>
<INPUT TYPE = "radio" NAME = "Q1" VALUE = "A" onclick
=Q1.value="A">
A.) Joint Deployable Intelligence Support Service
<BR>

<INPUT TYPE = "radio" NAME = "Q1" VALUE = "B" onclick = Q1.
value = "B">
B.) Joint Deployable Intelligence Support System
<BR>

<INPUT TYPE = "radio" NAME = "Q1" VALUE = "C" onclick =
Q1.value = "C">
C.) Joint Defense Information Sending System</FONT>
</P>

</OL>
```

```html
<P>
<HR>Question  #2:   &quot;Smart   push&quot;   compliments
&quot;pull&quot; architecture:</P>

<OL>
<P><INPUT TYPE = "radio" NAME = "Q2" VALUE = "A" onclick =
Q2.value = "A">
A). True<BR>
<INPUT TYPE = "radio" NAME = "Q2" VALUE = "B" onclick =
Q2.value="B">
B). False
<BR>
</OL>
</P>
<P>

<HR>Question #3: JDISS has been deployed in the following
exercises <B><I>except</I></B>:</P>

<OL>
<P><INPUT TYPE = "radio" NAME = "Q3" VALUE = "A"  onclick =
Q3.value = "A"> A). Desert Storm
<BR>
<INPUT TYPE = "radio" NAME = "Q3" VALUE = "B" onclick =
Q3.value = "B">B). Provide Promise
<BR>
<INPUT TYPE = "radio" NAME = "Q3" VALUE = "C" onclick =
Q3.value = "C" C). Restore Hope.
<BR>
</P>
</OL>

<P>
<HR>
<INPUT TYPE = "button" NAME = "submit" VALUE = "Grade Test"
onclick = scoretest(this.form)>
<INPUT   TYPE   =   "reset"   NAME   =   "reset"
VALUE="Reset"></FORM></P>

<H3>Correct Answers</H3>

<P>
<A HREF = "ans.html#a1" Target = "right"> #1 </A>, <A HREF =
"ans.html#a2" Target = "right"> #2 </A>,
<A HREF = "ans.html#a3" Target = "right"> #3 </A>
</P>

</BODY>
</HTML>
```

5. Review test JavaScript

```
!-- contents of index.html file -->

<HTML>
<HEAD>
<TITLE>Intro tc JDISS - Review test</TITLE>
</HEAD>

<SCRIPT LANGUAGE="JavaScript">
<!-- hide from old netscape browsers -->

// variables defined here may be accessed by any of the
frames defined

// below via the parent.variable_name command. NOTE: if the
NetScape window

// is resized, the variables will be reset to their initial
values.

// i.e., : these variables are available to the child frames.

// variables

var totalnum=3 // total # of questions

var correctans=0 // the question # (from 0 to N-1) of the
correct

// answer to each question. Set in Listing2 (scratcha.html)

var count=0 // counter variable. Initialized to zero

var arraycount=1 // index into the questans array. Intialized
to 1

// since that is where the first array string begins.

var totalright=0

// total # of questions answered right (0 initially)

var correcttext="blank"

// the text of the correct answer

<!-- done hiding from old netscape browsers -->
</SCRIPT>
```

```
<!-- must define two frames... although the second frame is
not used, a minimum of two frames is needed before netscape
will initiate a frame setup -->

<FRAMESET ROWS="*,0"> <!-- give the 2nd frame 0 pixels, and
assign the rest of the space to frame 1 via the "*" -->
     <FRAME SRC="listing2.html">
     <FRAME SCROLLING="no" NORESIZE>
</FRAMESET>

</HTML>
```

LIST OF REFERENCES

Allesi, Stephen M. And Trollip, Stanley R., *Computer-Based Instruction*, Prentice Hall, Inc., 1985.

Alsjuler, Liora, *ABCD...SGML*, International Thompson Computer Press, 1995, 2-22.

Bank, Joseph, "Java Security," http://swissnet.ai.edu/~jbank/Javapaper/Javapaper.html, 1996.

Booch, Grady, *Object Oriented Design*, Benjamin/Cummings Publishing Company, Inc., 1991.

Budd, Timothy, *Object Oriented Programming*, Addison-Wesley Publishing Co. Inc., 1991.

Conway, Melvin. E., "How Do Committees Invent?," *Datamation*, April 1968, 28-31.

Chappell, David, "CORBA 2.0: Will it Reach Far Enough?," *Network Collaboration*, March/April 1995, 16-18.

Crabb, Don, "Java Programming," *MacUser*, October 1996, 55-57.

Essex, W., *JDISS Migration to the DII*, JDISS PMO, Suitland, MD, 1996.

Flanagan, David, *JAVA in a Nutshell*, O'Reilly & Associates Inc., 1996, 6-10.

Flynn, Jim, "Battle for the Internet Infrastructure," *Datamation*, May 1, 1996, 28-35.

Gage, Deborah and Mardesich, Jodi, "ActiveX Standards Process Mired in Politics, Confusion," *Computer Reseller News,* September 9, 1996, 3, 215.

Gallagan, P. A. "Think Performance: A conversation with Gloria Gery," *Training and Development*, 1994, Vol. 48, No. 3, 47-51.

Gery, Gloria J., *Making CBT Happen: Prescriptions for Successful Implementation of Computer Based Training in Your Organization*, Ziff Communications Company, 1991, 6.

GITC, *Joint General Intelligence Training System Subarchitecture*, Vol 1., DIA 1994, 3-7 to 3-16.

Golberg, Adele and Rubin, Kenneth S., *Succeeding with Objects*, Addison-Wesley Publishing, 1995.

Goodman, Danny, *JavaScript Handbook*, IDG Books, 1996.

Guzdial and Kolodner, *Communications of the ACM*, April 1996, Vol 39, No. 4., 43-47.

Halfhill, Tom R., and Salamone, Salvatore, "Components Everywhere," *Byte*, January 1996, 97-104.

Jackson, Shari L., Stratford, Steven J., Krajcik, Joseph, and Soloway, Elliot, A Learner Centered Tool, *Communications of the ACM*, April 1996 Vol 39, No. 4, 48-49.

Jeiven, Helene, "A Common-Sense Checklist for CBT," *Training and Development*, July 1994, 47-49.

JDISS CCB, *JDISS Configuration Management Plan*, JDISS PMO, Suitland, MD, 1994, 1-13 .

JDISS PMO, *Concept of Operations for the Joint Deployable Intelligence Support System*, JDISS PMO, Suitland, MD, 1995, 1-4.

JDISS PMO, *Training Management Plan*, JDISS PMO, Suitland, MD, 1996, 1-8.

Kador, John, "The Ultimate Middeleware," *Datamation*, April 1996, 79-83.

Korzeniowski, Paul, "A Bridge Not Far?," *Open Systems Software Magazine*, June 1995, 103-109.

Kowitz, G. T. and Smith, J.C., "Three Forms of Instruction," *Journal of Educational Technology Systems*, 1987, 15(4), 419-429.

Lauzon, A. And Moore, G.A.B., "A Fourth Generation Distance Education System: Integrating Computer Assisted Learning and Computer Conferencing," *Distance Education for Corporate and Military Training*, Pennsylvania State University, 1992.

Lemay, Laura, and Perkins, Charles, et.al., *Teach Yourself JAVA*, Hayden Books, 1996, 491-503.

Linthicum, David S., "Integration, Not Perspiration," *Byte*, January 1996, 83-96.

Mowbray, Thomas J., and Zahavi, Ron, *The Essential CORBA*, John Wiley and Sons Inc., 1995.

McMahan, Paul D., *Technology Considerations for the Development of Multimedia Training at the Office of Naval Intelligence*, ONI, December 1995, 1-8.

MITRE Corporation, *A DODIIS Migration Strategy*, MITRE Corporation, McLean, VA 1996.

Naughton, Patrick, *The Java Handbook*, Osborne Mcgraw-Hill, 1996, 133-137.

Netscape, *http://www.netscape.com*, 1996.

Netscape Gold, *http://comprod/products/navigator/gold/datasheet.html*, 1996.

NIH, *http://rsb.info.nih.gov/nih-image/Java/Rotator*, 1996.

Object Management Group (OMG), *http://www.omg.org*, 1996a.

Object Management Group (OMG), *http://www.omg.org.pr96.comcorb.htm*, 1996b.

Orfali, R., Harkey, D., and Edwards, J. *The Essential Distributed Objects Survival Guide*, John Wiley and Sons Inc., 1996.

Reinhardt, Andy, "New Ways to Learn," *Byte*, March 1995, 50-71.

Russell, Deborah and Gangemi, G.T., Sr., *Computer Security Basics*, O'Reilly & Associates, 1991, 79-88.

Rymer, John R., "Objects can be in your future," *Network World*, January 16, 1995, 28.

Shackleton, John and Clark, John, "In Company Distance Learning," *Management Services*, March 1995, 24-25.

Soley, Richard, "Entering the Orbit of Object Request Brokers," *Network World*, April 10, 1995, 43.

SunSoft, *http://java.sun.com*, 1996.

Tibbitts, Fred, "CORBA: A common Touch for Distributed Applications," *Data Communications*, March 21, 1995, 71-75.

Weiland, Ross, "Staying in Touch," *Performance*, June 1995, 40-44.

INITIAL DISTRIBUTION LIST

No. of copies

1. Defense Technical Information Center   2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.   2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Professor Suresh Sridhar (Code SM/SR)   2
   Naval Postgraduate School
   Monterey, CA 93943-5002

4. Professor Tung Bui (Code SM/BD)   2
   Naval Postgraduate School
   Monterey, CA 93943-5002

5. CAPT Tom Dove, USN   4
   JDISS Program Management Office
   Office of Naval Intelligence
   Code: ONI-7JD
   4251 Suitland Road
   Washington, DC 20395-5720

6. LT John Capps, USN   1
   Fleet Intelligence Training Center Pacific
   33740 Puerto Rico St.
   San Diego, CA 92133-1847

7. CAPT William A. Kopper, USNR (ret)   1
   7050 Cloister Road
   Toledo, OH 43617

8. LT Bill Kopper, USN   2
   JDISS PMO
   Office of Naval Intelligence
   Code: ONI-7JD
   4251 Suitland Road
   Washington, DC 20395-5720